SDT-Models in R:
A Documentation of R-Code for Fitting Signal Detection Models

Dr. Siegfried Macho

University of Fribourg

Route de Faucigny 2

CH-1700 Fribourg,

Switzerland

email: siegfried.macho@unifr.ch

Date of last update: Friday, December 18, 2020

Table of Contents

1. Introduction

The SDT module consists of a set of functions that enable the fitting of various types of models to signal detection data. Parameters and asymptotic standard errors, as well as test statistics and model diagnostics are computed.

The entire module consists of two classes of source files:

1. *Source files with general functions for estimating parameters and for computing various statistics:*

`SDT-Main.R`	This file contains the main functions described below.
`SDT-Auxiliary.R`	This file contains auxiliary functions used by the main module.

2. *Source files containing the R-code of the different models:*

`SDT-SDT.R`	Standard Gaussian signal detection model
`SDT-MIX-PD.R`	Mixture model of three Gaussians for each pair of signal
`SDT-MIX-2.R`	Mixture model of two Gaussians for each signal (Generalized version of `SDT-MIX-PD`)
`SDT-DPSDT.R`	Dual process signal detection model (`DPSDT`)
`SDT-HT-n.R`	Double high threshold model for rating data
`SDT-SDT-2D.R`	Bivariate Gaussian signal detection model
`SDT2D-MIX-2.R`	Mixture model of two bivariate Gaussians for each signal (Bivariate version of `SDT-MIX-2`)
`SDT-mAFC.R`	Gaussian model for *m*-alternative forced choice task with bias
`SDT-Rank.R`	Gaussian SDT model for modeling ranking (*m*-AFC) and rating data.
`HTSDT-Rank.R`	Gaussian SDT model with a recollection and guessing component for modeling ranking (*m*-AFC) and rating data.
`HTSDT-Bias-Rank.R`	Gaussian SDT model with a recollection and guessing component for modeling ranking (*m*-AFC) and rating data with parameters representing position bias.
`MIX-Rank.R`	Gaussian mixture model for modeling ranking (*m*-AFC) and rating data.
`MIX-Bias-Rank.R`	Gaussian mixture model for modeling ranking (*m*-AFC) and rating data with parameters representing position bias.
`SDT-IRF-Gauss.R`	Gaussian SDT model with probabilistic item response functions.

The two files of the first class (`SDT-Main.R` and `SDT-Auxiliary.R`), comprising the basic estimation functions, have to be included into each application. The files of the second class contain the R-code of specific models. By consequence, only the file with the model actually

fitted has to be included. The inclusion of source files is performed by means of the R-function `source()` (see the examples in Chapters 5–15).
Alternatively, instead of using the `source()` function to include the source files, one my simply open each of the files that are required to perform the task (via the *File* menu option of the R console).

The following R-packages are required:

`numDeriv`	Contains functions for computing numerical gradients (Gilbert, 2006).
`ellipse`	Computation of bivariate confidence regions (required only for the bivariate models `SDT-SDT-2D` and `SDT2D-MIX-2`).
`mvtnorm`	Computes the distribution functions of the bivariate Gaussian distributions (required for the bivariate models `SDT-SDT-2D` and `SDT2D-MIX-2`).
`npmlreg`	The function `gqz()` of the package computes the points and weights for Gauss-Hermite quadrature (required for the models `SDT-mAFC.R`, `SDT-Rank.R`, `HTSDT-Rank.R`, `HTSDT-Bias-Rank.R`, `MIX-Rank.R`, `MIX-Bias-Rank.R` and `SDT-IRF-Gauss.R`).

A quick way to get acquainted with the functions of the modules consists in studying the working examples first (Chapters 5–16), and to consult the other chapters for further information. The different models are described in Chapter 3.

In case of questions, problems, and possible errors, respectively, please contact me.

2. Documentation of Principal Functions

The present chapter presents the main functions for estimating and evaluating models.

2.1 Estimation of parameters

Function call:

```
SDT.Estimate(par = NULL, data, Model.Id = "SDT", n = 2, fixed = NULL, ident
= NULL, functional = NULL, sym.gr = T, test = F, use.nlminb = T)
```

`par`	Vector with starting parameters. If no parameter vector is specified a vector of starting parameters is generated internally by the program. The order of parameters depends on the model to be estimated (cf., the descriptions of the single models in Chapter 2.5)
`data`	A data vector with observed frequencies: The data consist of response frequencies of the different response categories for each of type of signal presented.
`Model.Id`	A model identification string. For the moment the following options are available:
`"SDT"`	Standard signal detection model with $N(0, 1)$ as the noise distribution (= reference distribution) and a single set of decision bounds (cf. the description of the model in Chapter 3.1).
`"Gaussian"`	Signal detection model with different decision bounds for each of the different signal distributions. Without further constraints, this model is not identified. (cf. the description of the model in Chapter 3.2).

`"MIX.PD"`	Mixture model with each pair of signals (except for a possible additional foil signal) being represented by means of a mixture of three distributions (cf. the description of the model in Chapter 3.3).
`"MIX.2"`	Mixture model consisting of a mixture of two normal distributions per signal; Without further constraints, this model is not identified. (cf. the description of the model in Chapter 3.4).
`"DPSDT"`	Dual process signal detection model (cf. the description of the model in Chapter 3.5).
`"HT.n"`	Double high-threshold model for modeling rating data for pairs of signal (plus a possible additional foil signal) (cf. the description of the model in Chapter 3.6).
`"SDT.2D"`	Bivariate Gaussian signal detection model (cf. the description of the model in Chapter 3.7).
`"SDT2D.MIX.2"`	Mixture model consisting of a mixture of two bivariate normal distributions per signal (cf. the description of the model in Chapter 3.8).
`"mAFC"`	SDT model for *m*-alternative forced choice data (cf. the description of the model in Chapter 3.9).
`"SDT.Rank"`	SDT model for modeling forced choice (or ranking data) and rating data (cf. the description of the model in Chapter 3.10).
`"HTSDT.Rank"`	SDT model for modeling forced choice (or ranking data) and rating data, including a recollection and a guessing component (cf. the description of the model in Chapter 3.11.1).
`"HTSDT.Bias.Rank"`	SDT model for modeling forced choice (or ranking data) and rating data, including a recollection and a guessing component that enables the estimation of position bias (cf. the description of the model in Chapter 3.11.2).
`"MIX.Rank"`	SDT model for modeling forced choice (or ranking data) and rating data, using a mixture model (cf. the description of the model in Chapter 3.12.1).
`"MIX.Bias.Rank"`	SDT model for modeling forced choice (or ranking data) and rating data, using a mixture model that enables the estimation of position bias (cf. the description of the model in Chapter 3.12.2).

`n` Information about the model configuration whose structure depends on the model estimated:

1. Model `SDT`:
   A list with configuration information (for details, cf. Chapter 3.1).
2. Model `Gaussian`:
   A list with configuration information (for details, cf. Chapter 3.2).
3. Model `DPSDT`:
   A list with configuration information (for details, cf. Chapter 3.5).
4. Model `SDT.2D` and `SDT2D.MIX.2`:
   `n` is a list with information about the configuration of the model (for details, cf.

Chapter 3.7 and 3.8)

5. Model `mAFC`:
   `n` is a list with information about the configuration of the model. A detailed description of the entries of the list may be found in Chapter 3.9.
6. Model `SDT.Rank`:
   `n` is a list with information about the configuration of the model. A detailed description of the entries of the list may be found in Chapter 3.10.
7. Model `HTSDT.Rank` and `HTSDT.Bias.Rank`:
   `n` is also a list with information about the configuration of the model. The details of this list are described in Chapter 3.11.
8. Model `MIX.Rank` and `MIX.Bias.Rank`:
   `n` is also a list with information about the configuration of the model. The details of this list are described in Chapter 3.12.

`fixed` A 2×*p* matrix specifying the values of the *p* fixed parameters in the first and the positions of *p* fixed parameters within the parameter vector in the second row of the matrix (for further details, cf. Chapter 4.1).

`ident` A 2×*q* matrix specifying the positions of the source variables in the first row and positions of the target variables in the second row, for each of the of *q* equality constraints. The parameters whose positions are given in the second row are equated to the parameters whose positions are presented in the first row, in the same column (for further details, cf. Section 4.2).

`functional` A user defined function for specifying complex functional constraints on parameters (for further details, cf. Chapter 4.4).

`sym.gr` A flag indicating the use of a symbolic gradient function.
If `sym.gr = T` the function assumes a symbolic gradient function in case of functional constraints being present. In this case, a gradient function for the functional constraints has to be specified by the user (for further details, cf. Section 4.4).
If `sym.gr = F` a numerical approximation to the gradient function is performed in case of functional constraints being specified.

`test` If this flag is set to `TRUE` a numeric gradient of the log likelihood at the optimum is computed and compared to the symbolic gradient. The values of both gradients are printed (cf. the examples). The function `grad()` of the library `numDeriv` is used for computing the numeric gradient.

`use.nlminb` If `use.nlminb` = `T` (which is the default option) the `nlminb()` optimization routine is called first. This is followed by a call or the the optimizer `optim()` with the parameter vector resulting from the first estimation as start parameter.

Comment: The optimization routine `nlminb()` seems to be more stable than the routine `optim()`. However, `nlminb()` does not provide a Hessian matrix. Thus, a second run using `optim()` is performed.

Return value: An optimization object, that is, a list with the following fields:

1. `par` Estimated parameter vector
2. `value` The negative of the log likelihood function

3.	`counts`	Number of function evaluations
4.	`convergence`	0 = convergence reached
5.	`message`	Message indicating the stopping condition for the estimation process
6.	`hessian`	Observed information matrix
7.	`SDT.full.par`	The full parameter vector, including constraint or fixed parameters
8.	`SDT.par.names`	Names of the parameters that appear in the output
9.	`SDT.par.names.est`	Names of the free parameters that were actually estimated (these names appear in the output)
10.	`SDT.data`	Data vector
11.	`SDT.Model`	Model function for computing probabilities (cf. Chapter 2.5)
12.	`SDT.Model.Matrix.Anal`	Function computing the model matrix (i.e. matrix of partial derivatives of the model probabilities with respect to parameters) analytically, taking the constraints on parameters into account
13.	`SDT.Model.Matrix.Num`	Function computing the model matrix (i.e. matrix of partial derivatives of the model probabilities with respect to parameters) numerically, taking into account the constraints *Comment:* This functions is employed in case of no analytical function being available
14.	`SDT.Model.Id`	Model identification string, for specifying the model
15.	`SDT.n`	The number of different types of signals (signal distributions) or, alternatively a vector with the data points for the different types of signals
16.	`SDT.Model.Name`	A string with the model description that appears in the output
17.	`SDT.gr.sym.len`	Length of either the symbolic of the numeric gradient at optimum
18.	`SDT.fixed`	Matrix with fixed constraints
19.	`SDT.ident`	Matrix with equality constraints
20.	`SDT.functional`	Function specifying functional constraints
21.	`SDT.sym.gr`	Value of the flag indicating the computation of symbolic gradients.

Comments:

1. The first six entries are provided by the R function `optim()` whereas the residual entries, with the prefix `SDT` are added by the function `SDT.Estimate()`. The entries are used by the evaluation function `SDT.Statistics()` [cf. Chapter 2.2].
2. If functional constraints have been specified without a corresponding gradient function, the following error message is emitted in case of `sym.gr = T`:

```
Error in computation of gradient: No gradient for functional constraints found:
Set Parameter sym.gr = F in function SDT.Estimate !
```

   In this case, set `sym.gr = F` to enable the estimation with numerically computed derivatives.
3. Estimation of parameters without a symbolic gradient is considerably slower. However, to my knowledge, accuracy is not greatly affected.

Examples: cf. Chapter 5–15.

2.2 Computation of test statistics, standard errors of parameters, estimated probabilities and frequencies

Function call:

```
SDT.Statistics(Opti, deci = 3, conf = .95, display.warning = T, control =
NULL)
```

`Opti` — Optimization object (= the result of `SDT.Estimate()`)

`deci` — Number of decimal places in the output.

`Conf` — Size of the confidence intervals for estimated parameters.

`Display.warning` — Flag indicating whether a warning message should be displayed in case of a rank deficient Hessian matrix

`control` — A list with control information containing the following entries

- `LR` — Flag indicating whether to compute likelihood ratio based (LR) confidence intervals (`LR = T`)
- `factr` — A factor determining the size of the limits of the region where LR confidence bounds are searched: This region is defined by:

  estimate ± *factr*×(length of Wald confidence interval

  The default value is `factr = 2`.
- `Tol` — Tolerance level for computing LR confidence intervals (default: `1e-5`).
- `maxiter` — Maximal number of iterations for finding LR confidence bounds (default: `50`).
- `target.par` — A vector with the numbers of the parameters for which LR confidence intervals should be constructed (Numbers refer to the positions of parameters with respect to the full

parameter vector).

Return value: An object (i.e. a list) comprising the following fields:

1. *Model.description:* A string with a short description of the model
2. *Statistics:* A vector containing general statistical information with the following entries:
   - (i) Maximum of the log-likelihood function: log L
   - (ii) The Pearson X^2 statistic
   - (iii) The likelihood ratio statistic G^2
   - (iv) The degrees of freedom *df* associated with the Pearson and likelihood ratio statistic.
   - (v) The probability of $\chi^2(df) > X^2$
   - (vi) The probability of $\chi^2(df) > G^2$
   - (vii) Akaike's information criterion: $AIC = -2 \cdot \log L + 2 \cdot k$ (k = number of free parameters)
   - (viii) Bayesian information criterion: $BIC = -2 \cdot \log L + k \cdot \log N$ (N = total number of observations)
   - (ix) Consistent AIC with Fisher information (*CAICF*) [Bozdogan, 1987]:

     $$CAICF = -2 \cdot \log L + k \cdot (\log N + 2) + \log \left| \hat{\mathbf{I}} \right|$$

     ($\hat{\mathbf{I}}$ = observed information matrix; $\left| \hat{\mathbf{I}} \right|$ denotes the determinant of $\hat{\mathbf{I}}$)
   - (x) Scale dependent information complexity measure *ICOMP* (Bozdogan, 1988):

     $$ICOMP = -2 \cdot \log L + k \cdot \log \left[\frac{\text{trace}(\hat{\mathbf{\Sigma}})}{k} \right] - \log \left| \hat{\mathbf{\Sigma}} \right|$$

     ($\hat{\mathbf{\Sigma}}$ = estimated covariance matrix of parameters).
   - (xi) Scale invariant information complexity measure $ICOMP_R$:

     $$ICOMP_R = -2 \cdot \log L - \log \left| \hat{\mathbf{\Psi}} \right|$$

     ($\hat{\mathbf{\Psi}}$ = estimated correlation matrix of parameters).
   - (xii) Number of free parameters
   - (xiii) Length of the gradient at the optimum: should be very small (ideally 0)
   - (xiv) Rank of Hessian matrix (should correspond to number of free parameters)
   - (xv) The condition number of the observed information matrix (Hessian matrix): The condition number is defined as the ratio of the largest to the smallest non-zero singular value of the matrix.

     Comment: If the condition number is high (e.g. >5000), the results of the estimation process may not be stable, that is, small changes in the data may result in quite different estimates. In addition, some of the estimated standard errors of parameters may be quite high (cf. Chapter 5.6).
   - (xvi) Column rank of the model matrix $\mathbf{J}$ (by computing the rank of $\mathbf{J}^\mathbf{T} \cdot \mathbf{J}$). The rank should be equal to the number of free parameters.
   - (xvii) The condition number of $\mathbf{J}^\mathbf{T} \cdot \mathbf{J}$.

Comment: According to McDonald and Krane (1979) the matrix $\mathbf{J}^{\mathbf{T}} \cdot \mathbf{J}$ is better suited for detecting violations of local identification than the Hessian. This is confirmed by the example in Chapter 5.6 (and by each of the examples in the example files revealing problems of identification).

3. *Free.parameters:* Contains the estimated free parameters and, if the Hessian matrix is positive definite, the asymptotic standard errors (SE) of estimated parameters, as well as the lower and upper limits of the Wald confidence intervals.
   If wanted, likelihood ratio based confidence intervals are computed too (for the SDT and Gaussian model) [cf the examples in Chapter 5].

4. *Full.parametervector:* A *p*×*n* matrix of the *p* parameters (including both free and restricted ones) for the different types of signals and signal pairs, respectively.

   Comment:
   This section differs slightly for different types of models (see the examples in Chapter 5–15.

5. *SDT.measures:* A list of various SDT measures their standard errors (computed by means of the delta method) and confidence intervals.

   Comment: These measures are available with the model `"SDT"` only, in case of no functional constraints being specified.

d.a

$$d_a = \frac{\sqrt{2} \cdot d'_s}{\sqrt{\sigma_1^2 + \sigma_s^2}} = \frac{\sqrt{2} \cdot \mu_s}{\sqrt{1 + \sigma_s^2}}$$

where:

d'_s = Differences between the means of the Gaussian distributions, representing the first signal (whose mean is zero) and Signal *s*: $d'_s = \mu_s$

σ_1^2 = Variance of Gaussian distribution representing the first signal (=1)

σ_s^2 = Variance of Gaussian distribution representing the Signal *s*.

d.e

$$d_e = \frac{2 \cdot \mu_s}{1 + \sigma_s}$$

A_z

$$A_z = \Phi\left(\frac{d_a}{\sqrt{2}}\right) = \Phi\left(\frac{\mu_s}{\sqrt{1 + \sigma_s^2}}\right)$$ (area under the ROC curve)

where:

$\Phi(\ )$ represents the cumulative standard normal distribution

These measures are computed for each signal (other than the first signal) with respect to the first signal.

Principles underlying the computation of standard errors for specific SDT measures:

1. If both parameters (entering the measures) are fixed the computed standard error is zero.
2. If only one parameter is fixed the standard error of the non-fixed parameter enters the computation only.
3. In case of equality constraints the variances and covariances of the source parameter(s) enter(s) the computation.

6. Data.and.Estimates: A $n \cdot R_n \times 5$ matrix:

n	=	Number of signals (= signal distributions),
R_n	=	Number of data points (= response categories) per signal.

The columns contain the following information:

Column 1: Observed frequencies (data points)
Column 2: Estimated frequencies (frequencies estimated by the model)
Column 3: Estimated probabilities
Column 4: Pearson residuals computed by means of the following formula:

$$r_{ij} = \frac{f_{ij} - \hat{f}_{ij}}{\sqrt{\hat{f}_{ij}}},$$

where f_{ij} denotes the observed frequency of Response Category *I* for Signal *j* and $\hat{f}_{ij}$ symbolizes the respective estimated frequency.

Column 5: Standardized Residuals that are asymptotically N(0, 1) distributed.

The computation of standardized residuals is based on the following result (cf. Agresti, 2002, p.589):

$$\mathbf{r} \sim N\left(\mathbf{0}, \mathbf{I} - \left[\sqrt{\boldsymbol{\pi}_\mathbf{0}}\right] \cdot \left[\sqrt{\boldsymbol{\pi}_\mathbf{0}}\right]^\mathbf{T} - \mathbf{M} \cdot \left(\mathbf{M}^\mathbf{T} \cdot \mathbf{M}\right)^{-1} \cdot \mathbf{M}\right),$$

in words:

The vector of Pearson residuals $\mathbf{r}$ conforms to a normal distribution with mean $\mathbf{0}$ and covariance matrix (see above):

$$\boldsymbol{\Sigma} = \mathbf{I} - \left[\sqrt{\boldsymbol{\pi}_\mathbf{0}}\right] \cdot \left[\sqrt{\boldsymbol{\pi}_\mathbf{0}}\right]^\mathbf{T} - \mathbf{M} \cdot \left(\mathbf{M}^\mathbf{T} \cdot \mathbf{M}\right)^{-1} \cdot \mathbf{M}.$$

Thus, in order to compute the vector of standardized residuals the vector of Pearson residuals is multiplied by a diagonal matrix $\mathbf{D}(1/\boldsymbol{\sigma})$:

$$\mathbf{r}^{\text{standard}} = \mathbf{D}(1/\boldsymbol{\sigma}) \cdot \mathbf{r},$$

which amounts to dividing each element of $\mathbf{r}$ by the corresponding diagonal entry of $\boldsymbol{\Sigma}$.

Comment:

The matrix $\mathbf{H} = \left[\sqrt{\boldsymbol{\pi}_\mathbf{0}}\right] \cdot \left[\sqrt{\boldsymbol{\pi}_\mathbf{0}}\right]^\mathbf{T} - \mathbf{M} \cdot \left(\mathbf{M}^\mathbf{T} \cdot \mathbf{M}\right)^{-1} \cdot \mathbf{M}$ is the *hat matrix*, and $\boldsymbol{\Sigma} = \mathbf{I} - \mathbf{H}$.

By consequence, $r_i^{\text{standard}} = r_i / (1 - h_i)$, where r_i is the *i*-th entry of $\mathbf{r}$ and h_i is the *i*-th diagonal entry of the hat matrix $\mathbf{H}$.

The symbols have the following meaning:

$\mathbf{D}(1/\boldsymbol{\sigma})$	is a diagonal matrix with elements of the vector $1/\boldsymbol{\sigma} = 1/\sqrt{diag[\boldsymbol{\Sigma}]}$ in the main diagonal
$\mathbf{I}$	is the identity matrix.
$\sqrt{\boldsymbol{\pi}_0}$	is the vector of the square roots of probabilities predicted by the (true) model. For the actual computation this vector of probabilities is replaced by the probability vector given by the estimated model.
$\mathbf{M} = \mathbf{D}\left(1/\sqrt{\boldsymbol{\pi}_0}\right) \cdot \mathbf{J}$	is the scaled model matrix.
$\mathbf{J} = \frac{\partial \boldsymbol{\pi}}{\partial \boldsymbol{\theta}^{\mathbf{T}}}$	is the Jacobian matrix of the partial derivatives of the estimated probabilities $\boldsymbol{\pi}$ with respect to the model parameters $\boldsymbol{\theta}$.
$\mathbf{D}\left(1/\sqrt{\boldsymbol{\pi}_0}\right)$	is a diagonal matrix with elements $1/\sqrt{\boldsymbol{\pi}_0}$ on the main diagonal. Again, probabilities predicted by the esitamated model are used for the computation instead of the »true« probabilities in $\boldsymbol{\pi}_0$.

Example: cf. Chapter 5

Comments:

- If the Hessian matrix provided by the estimation routine is not positive definite a warning is printed together with the computed rank, if `display.warning = T`. In addition, instead of the standard deviations the values `NA` (*Not Available*) are printed.
- If the model matrix $\mathbf{J}$ is not of full column rank (evaluated by testing whether $\mathbf{J}^{\mathbf{T}} \cdot \mathbf{J}$ is positive definite) and if `display.warning = T` a warning is printed even if the Hessian matrix is positive definite. The matrix $\mathbf{J}^{\mathbf{T}} \cdot \mathbf{J}$ is usually a more reliable indicator of violations of local identification (cf. Chapter 5.6).
- If functional constraints are specified, the *analytic* computation of standardized residuals requires a function computing the Jacobian matrix of the functional constraints, i.e. the partial derivatives of the (constrained) parameters with respect to unconstrained parameters. This function must be passed to the function defining the functional constraints in the attribute `"SDT.Jacobian"` (cf. Chapter 4.4, and 5.4 for details).
  If the function computing the Jacobian matrix of the functional constraints is not provided or if the flag `sym.gr` has been set to `F` in the estimation procedure (thus preventing the computation of the symbolic gradient and model matrix during the estimation of parameters), the model matrix and the resulting standardized residuals are computed *numerically*.

2.3 Comparison of models

The following routine enables the comparison of likelihood ratio statistics for two models

Function call:

```
SDT.Anova(SO.1, SO.2, deci = 3)
```

`SO.1`	A statistics object (returned by the function `SDT.Statistics`) associated with Model 1
`SO.2`	A statistics object (returned by the function `SDT.Statistics`) associated with Model 2.
`deci`	Number of decimal places in the output.

Return value: A 3×3 matrix with each row containing the following entries:

Row 1 G^2 (likelihood ratio) statistic of Model 1 (first column).
Df (degrees of freedom) associated with G^2 of Model 1 (second column).
P value associated with G^2 and *df* for Model 1 (third column).

Row 2 G^2 (likelihood ratio) statistic of Model 2 (first column).
Df (degrees of freedom) associated with G^2 of Model 2 (second column).
P value associated with G^2 and *df* for Model 2 (third column).

Row 3 ΔG^2 (difference likelihood ratio) statistic (first column).
Df (difference in degrees of freedom associated with ΔG^2 (second column).
P value associated with ΔG^2 and *df* (third column).

Example: cf. Chapter 5.3.

2.4 Computing information about parameters as well as fixed and equality constraints

The following function computes a list containing the names of parameters as well as fixed and equality constraints. This function may be helpful for identifying the constraints specified for the model.

Function call:

```
SDT.Parameter.Info(data = NULL, par = NULL, n = 2, Model.Id = "SDT", fixed
= NULL, ident = NULL, deci = 3)
```

`data` A vector containing the data (not required if a parameter vector is passed for the second argument `par`).

`par` A parameter vector (not required if a data vector is passed as the first argument `data`).

`N` A list with configuration information or a number indicating the number of different types of signals.

`Model.Id` A model identification string (for possible strings, cf. Chapter 2.1).

`fixed` A 2×*p* matrix specifying the values of the *p* fixed parameters in the first and the positions of *p* fixed parameters within the parameter vector in the second row of the matrix (for further details, cf. Chapter 4.1).

`ident` A 2×*q* matrix specifying the positions of the source variables in the first row and positions of the target variables in the second row, for each of the of *q* equality constraints. The parameters whose positions are given in the second row are equated to the parameters whose positions are presented in the first row, in the same column (for further details, cf. Section 4.2).

`deci` Number of decimal places.

Comment: If no vector is passed, neither for `data` nor for `par`, the function gives an error message.

Return value: A list with two entries:

`$Model` A string denoting the model

`$Parameters.and.Constraints` A *p*×5 matrix with *p* = number of parameters and the five

columns providing the following information:

Column 1: `name` — The column contains the names of the parameters

Column 2: `par` — This columns contains the values of the parameters

Column 3: `fixed.value`
- ❒ If a parameter is fixed the column contains the specified fixed value.
- ❒ If the corresponding parameter is not fixed the string `---` is contained in the column.
- ❒ In case of a redundant probability that is fixed internally by the model (for example by the model `HT.n`) the column contains the string `Redundant-p`.

Column 4: `ident.source`
- ❒ If the parameter is subjected to an equality constraint (as a target) the name of the source of the equality constraint is contained in this column.
- ❒ In case of no equality constraint being specified for the parameter the string `---` is contained in the column.

Column 5: `nr` — If the parameter is subjected to an equality constraint (as a target) the number (=position) of the source of the equality constraint is given. Otherwise nothing is contained in the column.

Examples: cf. Chapters 5.1, 9.1, and 9.2.

2.5 Generating density and ROC plots

The following function may be used for plotting density and ROC curves of the model and the data.

Function call:

```
SDT.Plot(Object, cols = NULL, ltys = NULL, labels = NULL, SDT.legend =
NULL, option)
```

`Object` — An object containing the relevant information about the model. Two options are available:

1. The estimation object, i.e. the result of the function `SDT.Estimate` (cf. Chapter 2.1);
2. A list with the following entries:

   `Model.Id:` A model identification string; To date the following strings are valid:
   - ❒ `"SDT.2D"`
   - ❒ `"SDT"`
   - ❒ `"Gaussian"`
   - ❒ `"MIX.PD"`
   - ❒ `"MIX.2"`

   `par:` The full parameter vector;

   `n:` Number of signals or list with the configuration information (cf. Chapter 2.1)

   `counts:` A vector with data, i.e. the frequencies of different response categories [only relevant with option = `"ROC+"` or `"zROC+"` (see below)].

`cols` — A vector of length `s` (= number of signals) with numbers indicating colors of

the different models. The following colors are available, indicated by the numbers:

```
1 = "black",
2 = "grey70",
3 = "blue",
4 = "red",
5 = "cyan",
6 = "darkgreen",
7 = "magenta",
8 = "yellow"
```

If no colors are specified the models are drawn in black.

`ltys` A vector of length `s` (= number of signals) with numbers indicating the line types using the R convention:

```
1 = full line
2 = dotted
3 = dashed
```
(etc.).

If no line types are specified, all line are drawn with Line Type 1

`labels` A character vector of length 2 with the labels for the *x*- and *y*-axis

`SDT.legend` A list with the following entries:

- `pos` = position of the legend (if not specified, the legend is placed in the upper, left corner of the plot).

  If only the *x* coordinate is specified the *y* coordinate is computed by the program.
- `text` = vector of text strings.
- `cols` = colors of the single entries of the legend [if not specified, the colors from the main graphics are used (cf. above, option `cols`)].
- `ltys` = line types used for legend (if not specified, the line types of the main graphic is used

Comment:
The option SDT.legend is *not available* for the `SDT.2D` model

Example:

```
SDT.legend = list(pos = c(5, 0.4), text = paste("Signal", 1:5),
cols = 1:5)
```

`option`

`NULL`: Gaussian density curves are plotted (default option)

`"ROC"`: ROC curves are plotted, using the Gaussian distribution of the first signal as noise distribution

`"ROC+"` ROC curves plus data are plotted (data are either taken from the optimization object or must be provided by means of the entry data in the list with the data)

`"zROC"`: `z`ROC curves are plotted, using the Gaussian distribution of the first signal as noise distribution

`"zROC+"`: `z`ROC curves plus data are plotted (data are either taken from the optimization object or must be provided by means of the entry data in the list with the data)

In case of the SDT.2D model `option` may consist of a vector of numbers indicating those pieces of the decision bounds that are not plotted.

The order of the pieces of the bounds is shown in Figure 1. For example, with `option = c(1, 10)` the lower left horizontal and vertical pieces (with respective numbers) are not plotted (see also the example in Chapter 10.5).

Comments:

- A line width of 2 is used by default;
- The decision bounds are drawn in `grey80`;
- The function for plotting 90 percent confidence regions of the bivariate Gaussian distributions for the SDT.2D model requires the R-package `ellipse`.

Example 1:

```
Plot.Obj <- list(par = par.vec, n = n.data, Model.Id = "Gaussian")
SDT.Plot(Plot.Obj, cols = 3:7, labels = c("Memory Strength", "Density"))
```

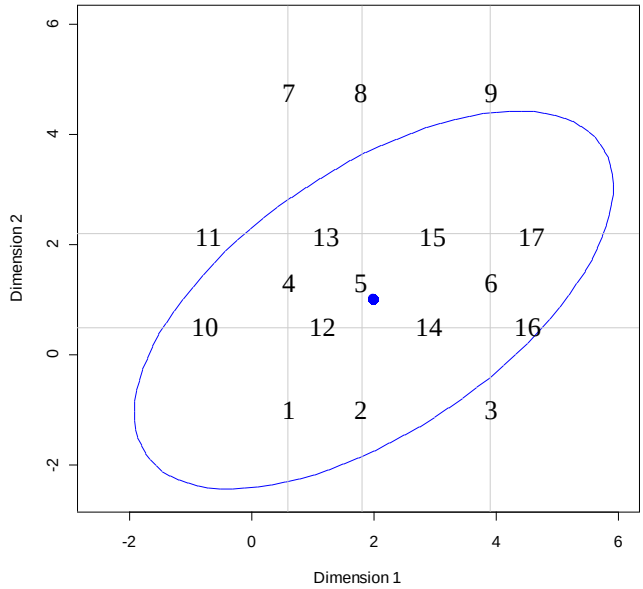


***Figure 1**: Order of decision bounds assumed by the argument `option` of the plot function..*

Example 2:

```
cfg <- list(s = 4, k1 = 5, k2 = 5, ds = 2)
xy.labels <- c("Low Frequency Signal", "High Frequency Signal")
Plot.Obj <- list(par = full.par, n = cfg, Model.Id = "SDT.2D")
SDT.Plot(Plot.Obj, cols = 3:6, labels = xy.labels)
```

Types of plots for different models:

`"SDT.2D"`	90 percent confidence ellipses for the SDT-2D model are plotted as well as the decision bounds (cf. Figure 17, Figure 18, and Figure 19).

`"SDT"`
`"Gaussian"`
`"MIX.PD"`
`"MIX.2"`
	Density curves with decision bounds, ROC curves or zROC curves with and without data are plotted.

2.6 Functions for computing the area under the empirical ROC

The following functions can be used to compute the area und the ROC (receiver operating characteristic). The functions are located in the file `SDT-Auxiliary.R`.

2.6.1 Area under the empirical ROC curve using the trapezoid rule (Macmillan & Creelman, 2005):

Function call:

```
ROC.Ag(FA, H, show.p = F, reverse = F)
```

`FA`	Vector containing the frequencies of response categories for new / noise etc. items, with response categories ordered from *sure new/noise* to *sure old/signal*.
`H`	Vector containing the frequencies of response categories for old / signal etc. items, with response categories ordered from *sure new/noise* to *sure old/signal*.
`show.p`	Flag indicating whether intermediate results should be shown (`show.p = TRUE`)
`reverse`	Flag indicating that response categories are ordered from *sure old/signal* to *.sure new/noise.*

Output:
The value of the area computed by the trapezoid rule.

Example (Macmillan & Creelman, 2005, p.57):

```
new.items <- c(30, 23, 37,  8,  2)
old.items <- c( 4,  5, 15, 15, 61)
ROC.Ag(new.items, old.items, T)
```

Result:

```
0.8789
```

Comment: Cf. the example file: `A.g (Area under the ROC trapezoidal).R`

2.6.2 Area under the empirical ROC curve using the method of Donaldson & Good (1996):

Function call:

```
ROC.Ar(FA, H, probs = F, reverse = F, coll = T))
```

`FA`	Vector containing the frequencies of response categories for new / noise etc. items, with response categories ordered from *sure new/noise* to *sure old/signal*.
`H`	Vector containing the frequencies of response categories for old / signal etc. items, with response categories ordered from *sure new/noise* to *sure old/signal*.
`probs`	Flag indicating whether probabilities are used as input (in the first two arguments) instead of frequencies.
`reverse`	Flag indicating that response categories are ordered from *sure old/signal* to *sure new/noise.*
`coll`	Flag indicating whether points on the ROC are collapsed (eliminated) in case of improper ROCs.

Output:
A list with two elements:

(1) A.r = The area A'r according to Donaldson & Good (1996)
(2) npt = Number of ROC points used for the computation.

Example (Donaldson & Good, 1996, p. 593):

```
new.items <- c(19, 31, 17, 32, 16, 13)
old.items <- c( 5, 12,  9, 20, 23, 59)
ROC.Ar(new.items, old.items)
```

Result:

```
$A.r
[1] 0.7628946
$npt
[1] 5
```

Comment: Cf. the example file: `A.r (Area under ROC Donaldson & Good).R`

2.6.3 Area under the Gaussian ROC using principal component analysis (Vokey, 2016):

Function call:

```
ROC.PCA(FA, H, reverse = F, deci = NULL, correct = T)
```

`FA`	Vector containing the frequencies of response categories for new / noise etc. items, with response categories ordered from *sure new/noise* to *sure old/signal*.
`H`	Vector containing the frequencies of response categories for old / signal etc. items, with response categories ordered from *sure new/noise* to *sure old/signal*.
`reverse`	Flag indicating that response categories are ordered from *sure old/signal* to *sure new/noise.*
`deci`	Number of decimal places used for result (if `deci = NULL` no rounding is performed).
`correct`	Use Laplace's rule of succession for correction of possibly empty cells.

Output:
A list comprising the following elements:

(1) slope = slope of the first principal axis of the *z*-ROC,
(2) intercept = intercept of the first principal axis of the *z*-ROC,
(3) d.1 = d1 of Macmillan & Creelman (2005),
(4) d.2 = d2 of Macmillan & Creelman (2005),
(5) d.a = sensitivity measure d_a of Simpson & Fitter (1973),
(6) d.e = sensitivity measure d_e of Simpson & Fitter (1973),
(7) d.p = a new sensitivity measure of Vokey (2016),
(8) d.YNp = a new measure proposed by Vokey (2016),
(9) A.z = area under Gaussian ROC,
(10) A.zp = an alternative area measure based on d.YNp
(11) explained = variance explained by the first principal component.

Example (Vokey, 2016, p.10):

```
new.items <- c(334, 99, 26, 42, 43, 47)
old.items <- c(81, 44, 26, 32, 66, 348)
ROC.PCA(new.items, old.items)
```

Result:

```
$slope
[1] 0.7172354
$intercept
[1] 1.228804
$d.2
[1] 1.228804
$d.1
[1] 1.71325
$d.a
[1] 1.412126
$d.e
[1] 1.326119
$d.p
[1] 1.453928
$d.YNp
[1] 1.028082
$A.z
[1] 0.8409872
$A.zp
[1] 0.8480444
$explained
[1] 0.9993234
```

Comment: Cf. the example file: `PCA of ROC.R`

2.6.4 Area under ROC computed by the trapezoid rule plus an estimate of the standard error (Hanley & McNeil, 1982).

Function call:

```
Area.HanleyMcNeil(FA, H, show = F, neg.exp = F)
```

`FA`	Vector containing the frequencies of response categories for new / noise etc. items, with response categories ordered from *sure new/noise* to *sure old/signal*.
`H`	Vector containing the frequencies of response categories for old / signal etc. items, with response categories ordered from *sure new/noise* to *sure old/signal*.
`show`	Flag indicating whether intermediate results should be displayed.
`neg.exp`	Flag indicating whether a simplified formula assuming a negative exponential model should be used for computing the estimated standard error of the area.

Output:

A list with two elements:

(1) W = area under the ROC computed by means of the trapezoid rule (that corresponds to the Wilcoxon statistic W);

(2) SE = estimated standard error of the estimated area.

Example (Hanley & McNeil, 1982):

```
normal   <- c(33, 6, 6, 11, 2)
abnormal <- c(3, 2, 2, 11, 33)
Area.HanleyMcNeil(normal, abnormal)
```

Result:

```
$W
[1] 0.8931711
$SE
[1] 0.03199041
```

Comment: Cf. the example file: `Area (Hanley-McNeil).R`

2.6.5 Function for computing the standard error of the area under the empirical ROC (determined by means of the trapezoid rule) [Hanley & McNeil, 1982]:

Function call:

```
SE.Area.HanleyMcNeil(W, n.new, n.old)
```

`W`	Area under the empirical ROC, determined by means of the trapezoid rule.
`n.new`	Number of new / noise etc. trials.
`n.old`	Number of old / signal etc. trials.

Output:
The computed standard error of the area.

Example (Hanley & McNeil, 1982):

```
Area  <- 0.85
n.new <- 40
n.old <- 40
SE.Area.HanleyMcNeil(Area, n.new, n.old)
```

Result:

```
0.04373749
```

Comment: Cf. the example file: `Area (Hanley-McNeil).R`

3. Description of the models

3.1 The standard Gaussian signal detection model (SDT)

(1) Model structure:
See any text on signal detection models (e.g., Macmillan & Creelman, 2005; Wickens, 2002).
The noise distribution is $N(0, 1)$, i.e., the standard normal distribution (with mean zero and variance 1.0).

(2) Model identification string: `"SDT"`
This string can be passed to the function `SDT.Estimate()` with the argument `Model.Id` (cf. Chapter 1). Due to the fact that the standard SDT model is the default model, no argument has to be passed in case of using the standard SDT model.

(3) Name of the file containing the model:
The source code of the model is contained in the file: `SDT-SDT.R`.

(4) Configuration information:

The configuration information is passed to the function `SDT.Estimate()` in the argument `n`. The configuration information consists of a list with two entries:

`n.sdt` = The number of signals (default: `n.sdt = 2`)

`restriction` = A string specifying the type of restrictions: The following options are available:

`"no"`	No restrictions (=default)
`"equalvar"`	The variance parameters of all Gaussian distributions are set equal to 1.0.

Example:

```
n <- list(n.sdt = 4, restriction = "equalvar")
```

tells the estimation function that there are 4 types of signals and equality constraints on variance parameters. By consequence the equal variance signal detection model with four Gaussian models is fitted (see also the examples in Chapter 5.1).

Comment:

Partial matching of the string works also, for example, `restriction = "equal"` or `restriction = "EQUALVAR"` lead to the same outcome.

(5) Order of parameters:

Parameters of the model are in the following order (passed to the function `SDT.Estimate()` in the parameter `par`):

I. Two parameters characterizing the Gaussian signal distributions repeated for each signal distribution, except for the noise distribution N(0, 1) whose parameters are fixed.

(i) μ_j = Mean of the Gaussian model representing signal distribution *j*.

(ii) σ_j = Standard deviation of the Gaussian model representing signal distribution *j*.

Comment:

For the standard SDT model $j = 2,3,...,n$, since the mean and variance parameter of the noise distribution ($j = 1$) are fixed and need not be specified.

II. $t_1, t_2, ..., t_{R-1}$, = thresholds (decision bounds), where *R* is the number of response categories.

Help/Tip:

The function:

```
SDT.Parameter.Info(data = NULL, par = NULL, n = 2, Model.Id =
"SDT", fixed = NULL, ident = NULL, deci = 3)
```

displays the parameter configuration (a description of the parameters of the function is given in Chapter 2.4):

Example:
Given: The data of Ratcliff et al. (1994), Experiment 1, pure lists (4 types of signals with 6 response categories per signal):

```
datavec <- c(477, 776, 527, 321, 258, 184,      # Pure Strong New
             192, 401, 290, 267, 316, 442,      # Pure Strong Old
             235, 649, 719, 442, 254, 156,      # Pure Weak   New
             151, 496, 480, 350, 221, 142)      # Pure Weak   Old
```

The sequence of commands:

```
n <- list(n.sdt = 4, restriction = "equal")
PI <- SDT.Parameter.Info(data = datavec, n = n)
print(PI)
```

results in the following output:

```
[1] "Standard SDT model with noise distribution N(0, 1), Number of
models: <4>, Type of restrictions: <equalvar>"

$Parameters.and.Constraints
        name   par fixed.value ident.source Nr
1    Mean[2]  0.00         ---          ---
2  Stddev[2]  1.00     1 <set>          ---
3    Mean[3]  0.00         ---          ---
4  Stddev[3]  1.00     1 <set>          ---
5    Mean[4]  0.00         ---          ---
6  Stddev[4]  1.00     1 <set>          ---
7         t-1 -0.90         ---          ---
8         t-2 -0.45         ---          ---
9         t-3  0.00         ---          ---
10        t-4  0.45         ---          ---
11        t-5  0.90         ---          ---
```

Comments:

- Note that no mean and standard deviation is provided for the first signal. These parameters are assumed to have fixed values (0 and 1, respectively).
- Due to the specification of fixed constrains on variance parameters these parameters are set equal to 1.0 as indicated by the symbol `<set>`.

(6) Order of input data:
The input data are response frequencies for the different response categories. The data for the noise signal are presented first, followed by the data for the other signals. The order of the data within each signal is from the highest noise (new) category (e.g. "sure noise" or "sure new") to the highest signal (old) category (e.g. "sure signal" or "sure old").

Comment: The output presents the data in the same order.

(7) Model function:
The function for computing model probabilities:

```
SDT.SDT(par, n)
```

`par` Vector of parameters

`n` List with configuration information (see above).

Comment: The function can be used in isolation, e.g. for generating artificial data.

(8) Model matrix (analytical computation):
The model matrix (i.e., a $m \times p$ matrix of partial derivatives of the model probabilities with respect to the free parameters [m = number of generated probabilities (= number of data points), p = number of free model parameters] is computed by the following function:

```
SDT.SDT.Model.Matrix(par, n, fixed = NULL, ident = NULL, functional =
NULL)
```

`par`	Vector of parameters
`n`	List with configuration information (see above).
`Fixed`	$2 \times q$ matrix of fixed constraints (cf. Chapter 4.1)
`ident`	$2 \times r$ matrix of equality constraints (cf. Chapter 4.2)
`functional`	Function implementing functional constraints. The function has to contain the attribute `"SDT.Jacobian"` containing a function for computing the Jacobian matrix of the functional constraints. If this function is not available in case of functional constraints being specified, the function computing the model matrix returns the value `NULL`. (For further details, see Chapter 4.4)

Comment: The function can be used in isolation, e.g. for specific tests of the model.

(9) Model matrix (numerical computation):

The following function computes the model matrix numerically, using the function `jacobian()` from the package `NumDeriv`.

```
SDT.SDT.Model.Matrix.Num(par, n, fixed = NULL, ident = NULL, functional
= NULL)
```

`par`	Vector of parameters
`n`	List with configuration information (see above).
`Fixed`	$2 \times q$ matrix of fixed constraints (cf. Chapter 4.1)
`ident`	$2 \times r$ matrix of equality constraints (cf. Chapter 4.2)
`functional`	Function implementing functional constraints.

Comments:

- Contrary to the analytical counterpart this function does not require the specification of a function computing the Jacobian matrix in case of functional constraints being specified.
- The function is invoked for computing the model matrix in the function `SDT.Statistics()` in the following situations:
  - (i) Functional constraints have been specified by the user without specifying a function for computing the Jacobian of the constraints.
  - (ii) The flag `sym.gr` was set to `FALSE` in the estimation procedure thus preventing the computation of symbolic derivatives.

(10) Hessian matrix:

The following function computes the Hessian matrix analytically:

```
SDT.Hessian(parvec, datavec, n.s, fixed = NULL, ident = NULL, functional
= NULL)
```

`parvec`	Vector of parameters
`datavec`	Data vector
`n.s`	List with configuration information (see above).
`Fixed`	$2 \times q$ matrix of fixed constraints (cf. Chapter 4.1)
`ident`	$2 \times r$ matrix of equality constraints (cf. Chapter 4.2)
`functional`	Function implementing functional constraints.

(11) Examples:

See Chapter 5.1.1

3.2 The Gaussian signal detection model with the full set of free parameters (Gaussian)

(1) Model structure:
See any text on signal detection models (e.g., Macmillan & Creelman, 2005; Wickens, 2002).

(2) Model identification string: `"Gaussian"`

(3) Name of the file containing the model:
The source code of the model is contained in the file: `SDT-SDT.R`.

(4) Configuration information:
The configuration information is passed to the function `SDT.Estimate()` in the argument `n`. The configuration information consists of a list with two entries:

`n.sdt` = 1. The number of signals (default: `n.sdt = 2`), or
2. A vector with the number of data points for each type of signal. This enables the fitting of different numbers of data points for different signals (cf. Chapter 5.5).

`restriction` = A string specifying the type of restrictions. The following options are available:

`"no"`	No restrictions are specified (the model with this specification is not identified).
`"standard"`	Restrictions conforming to the standard unequal variance detection model with noise distribution: $N(0, 1)$. Thus the mean and variance of the first Gaussian distribution are set equal to 0.0 and 1.0 and the threshold parameters assigned to the different Gaussian distributions are constrained to be equal. As a result, the model with this option is identical to the `SDT` model (cf. Chapter 3.1). This restriction is the default option.
`"symmetric"`	Instead of setting the mean of the first distribution to 0.0 the mean of the first Gaussian distribution (the noise distribution) is set minus to the mean of the second distribution: $\mu_1 = -\mu_2$.
`"equalvar"`	The variance parameters of all Gaussian distributions are set equal to 1.0, additionally to the standard restrictions.
`"equalvar-symmetric"` `"equal-symmetric"`	The variance parameters of all Gaussian distributions are set equal to 1.0, additionally to the symmetric restrictions.
`"pairs"`	The following restrictions are set: 1. Gaussian distribution of model 1, 3, 5, … are set to $N(0, 1)$. 2. The decision bounds of Model 1 are set equal to that of Model 2, those of Model 3 are set equal to those of Model 3 etc. (always pairwise).

`"pairs-equalvar"`	The following restrictions are set: 1. Mean parameters of the Gaussian distribution of model 1, 3, 5, … are set to zero. 2. All variance parameters are set to 1.0. 3. The decision bounds of Model 1 are set equal to that of Model 2, those of Model 3 are set equal to those of Model 3 etc. (always pairwise).
`"standard-pairs"`	The following restrictions are set: 1. Gaussian distribution of model 1, 3, 5, … are set to $N(0, 1)$. 2. The decision bounds of the different models are assumed to be equal.
`"standard-pairs-equalvar"`	The following restrictions are set: 1. Mean parameters of the Gaussian distribution of model 1, 3, 5, … are set to zero. 2. All variance parameters are set to 1.0. 3. The decision bounds of the different models are assumed to be equal.

Examples:

```
n <- list(n.sdt = 2, restriction = "equal-symmetric")
```

tells the estimation function that there are 2 types of signals and equal variance symmetric restrictions should be set. By consequence, the equal variance signal detection model with two Gaussian models is fitted, with the restriction: $\mu_1 = -\mu_2$ (cf. the example in Chapter 5.1.2).

Comments:

- Partial matching of the string works also, for example, `restriction = "equal-sym"` or `restriction = "EQUAL-SYM"` lead to the same outcome.
- A demonstration of various restrictions may be found in the example source file: `SDT-Gauss (Parameter Information).R`

(5) Order of parameters:

The parameters of the model are in the following order (passed to the function `SDT.Estimate()` in the parameter `par`):

For each of the $j = 1,2,...,n$ signal distribution the parameters are given in the following order:

(i) μ_j = Mean of the Gaussian model representing signal distribution *j*.

(ii) σ_j = Standard deviation of the Gaussian model representing signal distribution *j*.

(iii) $t_{j1}, t_{j2}, ..., t_{j,R_j-1}$, = thresholds (decision bounds), where R_j = the number of responses for signal *j*.

Comments:

(i) Contrary to the standard SDT model, the full set of parameters as well as the threshold values have to be specified for each type of signal.

(ii) The Gaussian model enables the fitting of different number of data points for the different types of signals.

Help/Tip:

The function:

```
SDT.Parameter.Info(data = NULL, par = NULL, n = 2, Model.Id =
"Gaussian", fixed = NULL, ident = NULL, deci = 3)
```

displays the parameter (a description of the parameters of the function is given in cf. Chapter 2.4):

Example:

Given: The data of Ratcliff et al. (1994), Experiment 1, pure lists (2 types of signals with 6 response categories per signal):

```
datavec <- c(477, 776, 527, 321, 258, 184,      # Pure Strong New
             192, 401, 290, 267, 316, 442,      # Pure Strong Old
             235, 649, 719, 442, 254, 156,      # Pure Weak   New
             151, 496, 480, 350, 221, 142)      # Pure Weak   Old
```

The sequence of commands:

```
n <- list(n.sdt = 4)
PI <- SDT.Parameter.Info(data = datavec, n = n, Model.Id =
"Gaussian")
print(PI)
```

results in the following output:

```
$Model
[1] "Gaussian SDT model with freely estimable parameters for each
model, Number of models: <4>, Type of restrictions: <standard>"

$Parameters.and.Constraints
        name   par fixed.value ident.source Nr
1    Mean[1]  0.00     0 <set>          ---
2  Stddev[1]  1.00     1 <set>          ---
3     t-1[1] -1.50         ---          ---
4     t-2[1] -0.75         ---          ---
5     t-3[1]  0.00         ---          ---
6     t-4[1]  0.75         ---          ---
7     t-5[1]  1.50         ---          ---
8    Mean[2]  0.00         ---          ---
9  Stddev[2]  1.00         ---          ---
10    t-1[2] -1.50         --- t-1[1] <set>  3
11    t-2[2] -0.75         --- t-2[1] <set>  4
12    t-3[2]  0.00         --- t-3[1] <set>  5
13    t-4[2]  0.75         --- t-4[1] <set>  6
14    t-5[2]  1.50         --- t-5[1] <set>  7
15   Mean[3]  0.00         ---          ---
16 Stddev[3]  1.00         ---          ---
17    t-1[3] -1.50         --- t-1[1] <set>  3
18    t-2[3] -0.75         --- t-2[1] <set>  4
19    t-3[3]  0.00         --- t-3[1] <set>  5
20    t-4[3]  0.75         --- t-4[1] <set>  6
21    t-5[3]  1.50         --- t-5[1] <set>  7
22   Mean[4]  0.00         ---          ---
23 Stddev[4]  1.00         ---          ---
24    t-1[4] -1.50         --- t-1[1] <set>  3
25    t-2[4] -0.75         --- t-2[1] <set>  4
26    t-3[4]  0.00         --- t-3[1] <set>  5
27    t-4[4]  0.75         --- t-4[1] <set>  6
28    t-5[4]  1.50         --- t-5[1] <set>  7
```

Comments:

- Due to the fact that no restrictions have been specified standard restrictions which are the default are provided by the model. This is indicated by the symbol `<set>`. On the lines with the mean `Mean[1]` and standard deviation `Stddev[1]` of the first model.
- The equality constraints are indicated by the entries of the last columns. For example the entry for Parameter 10:

  ```
  10    t-1[2] -1.50          --- t-1[1] <set>  3
  ```

  indicates that first threshold assigned with the second Gaussian model `t-1[2]` was set equal to first threshold of the first Gaussian model `t-1[1]` which corresponds to Parameter 3.

(6) Order of data:

The input data are response frequencies for the different response categories. The data for the noise signal are presented first, followed by the data for the other signals. The order of the data within each signal is from the highest noise (new) category (e.g. "sure noise" or "sure new") to the highest signal (old) category (e.g. "sure signal" or "sure old").

Comment: The output presents the data in the same order.

(7) Model function:

The function for computing model probabilities:

```
SDT.Gaussian(par, n)
```

`par` Vector of parameters

`n` List with configuration information (see above).

Comment: The function can be used in isolation, e.g. for generating artificial data.

(8) Model matrix (analytical computation):

The model matrix (i.e., a $m \times p$ matrix of partial derivatives of the model probabilities with respect to the free parameters [m = number of generated probabilities (= number of data points), p = number of free model parameters] is computed by the following function:

```
SDT.Gaussian.Model.Matrix(par, n, fixed = NULL, ident = NULL, functional
= NULL)
```

`par`	Vector of parameters
`n`	List with configuration information (see above).
`fixed`	$2 \times q$ matrix of fixed constraints (cf. Chapter 4.1)
`ident`	$2 \times r$ matrix of equality constraints (cf. Chapter 4.2)
`functional`	Function implementing functional constraints. The function has to contain the attribute `"SDT.Jacobian"` containing a function for computing the Jacobian matrix of the functional constraints. If this function is not available in case of functional constraints being specified, the function computing the model matrix returns the value `NULL`. (For further details, see Chapter 4.4)

Comment: The function can be used in isolation, e.g. for specific tests of the model.

(9) Model matrix (numerical computation):

The following function computes the model matrix numerically, using the function `jacobian()` from the package `NumDeriv`.

```
SDT.Gaussian.Model.Matrix.Num(par, n, fixed = NULL, ident = NULL,
functional = NULL)
```

`par`	Vector of parameters
`n`	List with configuration information (see above).
`Fixed`	$2 \times q$ matrix of fixed constraints (cf. Chapter 4.1)
`ident`	$2 \times r$ matrix of equality constraints (cf. Chapter 4.2)
`functional`	Function implementing functional constraints.

Comments:

- Contrary to the analytical counterpart this function does not require the specification of a function computing the Jacobian matrix in case of functional constraints being specified.
- The function is invoked for computing the model matrix in the function `SDT.Statistics()` in the following situations:
  (i) Functional constraints have been specified by the user without specifying a function for computing the Jacobian of the constraints.
  (ii) The flag `sym.gr` was set to `FALSE` in the estimation procedure thus preventing the computation of symbolic derivatives.

(10) Example:

See Chapter 5.

3.3 The mixture model with each pair of signals represented by a mixture of three Gaussian distributions (MIX.PD)

The mixture model enables one to model source monitoring data, like those of Hilford, Glanzer, Kim, & DeCarlo (2002) or DeCarlo (2003a), as well as more complex data sets like those of Kelley & Wixted (2001) on associative recognition (cf. Chapter 5.7).

(1) Model structure:
Figure 2 depicts the structure of the model for one pair *j* of signals.

The parameter $\pi_{j_{\text{left}}}$ represents the probability of using the "left" Gaussian distribution $N\left(\mu_j - d'_{j_{\text{left}}}, \sigma_j^2\right)$ for modeling the response frequencies for the first signal of the pair, whereas with probability $1-\pi_{j_{\text{left}}}$ the "middle" distribution $N\left(\mu_j, \sigma_j^2\right)$ is employed.

The parameter $\pi_{j_{\text{right}}}$ represents the probability of using the "right" Gaussian distribution $N\left(\mu_j + d'_{j_{\text{right}}}, \sigma_j^2\right)$ for modeling the response frequencies for the second signal of the pair, whereas with probability $1-\pi_{j_{\text{right}}}$ the "middle" distribution $N\left(\mu_j, \sigma_j^2\right)$ is employed.

This structure holds for each pair j $\left(j = 1,2,...,J\right)$ of stimuli.

Comment: The resulting mixture of densities is shown in Figure 3 (using parameter estimates for the data of Hilford et al. (2002), Exp.2):

(2) Model identification string: `"MIX.PD"`

(3) Name of the file containing the model:
The source code of the model is contained in the file: `SDT-MIX-PD.R`.

(4) Order of parameters:
The parameters of the model are passed to the function `SDT.Estimate()` in the parameter `par` in the following order:

A. For each pair of signals (except for the first pair in case of no foil distribution being present) the following 6 parameters have to provided in the following order:

(i) μ_j = Mean of the "middle" Gaussian distribution for signal pair *j*.

(ii) σ_j = Standard deviation of the "middle" as well as the "left" and "right" Gaussian distributions for signal pair *j*.

(iii) $\pi_{j_{\text{left}}}$ = The probability of invoking the "left" Gaussian distribution for the first signal of signal pair *j*.

(iv) $\pi_{j_{\text{right}}}$ = The probability of invoking the "right" Gaussian distribution for the second signal of signal pair *j*.

(v) $d'_{j_{\text{left}}}$ = Displacement of the "left" distribution with respect to the "middle" for the first signal of signal pair *j* (cf. Figure 2).

(vi) $d'_{j_{\text{right}}}$ = Displacement of the "right" distribution with respect to the "middle" for the second signal of signal pair *j* (cf. Figure 2).

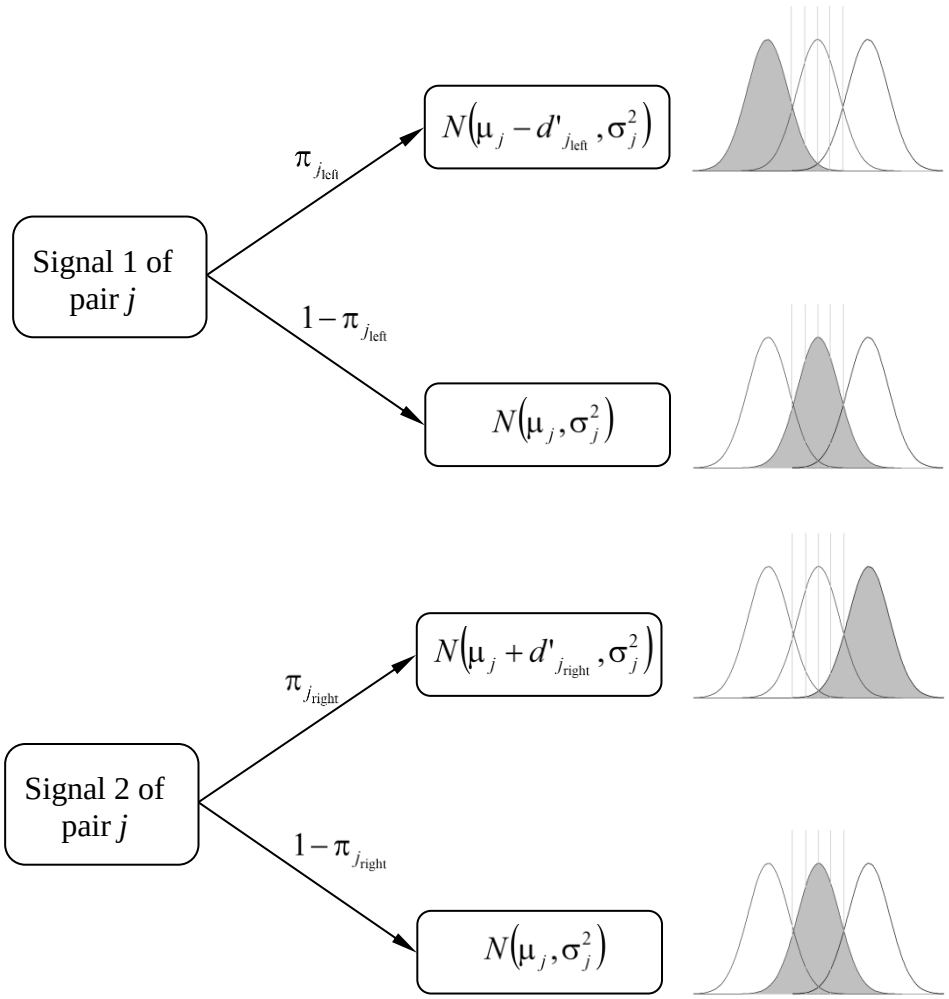


Figure 2*: Structure of the MIX.PD model (mixture of three Gaussian distributions for each pair of signals).*

B. After specification of the parameters for each signal pair the threshold parameters have to be specified:

$t_1, t_2, ..., t_{R-1}$, = thresholds (decision bounds), where R is the number of response categories per signal (assumed to be the same for each signal).

There are two different cases that have to be considered:

(i) An *even* number of signals is modeled: In this case, the mean and variance of the middle Gaussian distribution for the first signal is assumed to be 0 and 1, respectively. By consequence, the mean and the variance parameters for the first pair of signals must not to be specified.

(ii) An *uneven* number of signals is given: In this case, the procedure assumes that the first signal represents *foils*. The location and scale of the whole configuration is fixed by assuming the mean and standard deviation of foil distribution to be 0 and 1, respectively.

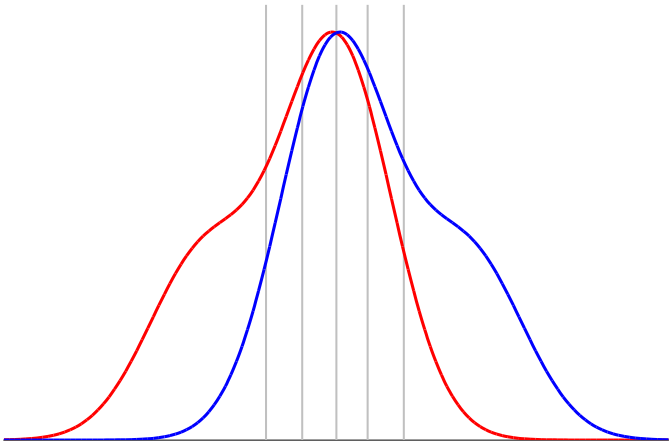


***Figure 3**: Mixture of densities based on the results of Hilford et al. (2002), Experiment 2*

Help/Tip:

The function `SDT.MIX.PD.start.par(n, data)` creates starting parameters for the given number of signals `n` and the available data `data`.
The function `MIX.PD.par.names(par, n)` creates names of the parameters in `par` for the number of signals `n` (that are also shown in the output).

Example:

Given: The data of Kelley and Wixted (2001), Experiment 1 (Two pairs of signals with 6 response categories per signal):

```
data <- c(198, 148, 210,  85,  49,  30,   # Weak   Items, rearranged
           72,  91, 167,  71,  71, 248,   # Weak   Items, intact
          305, 110, 126,  79,  45,  55,   # Strong Items, rearranged
           19,  23,  36,  57,  81, 504)   # Strong Items, intact
```

The sequence of commands:

```
n <- 4
par <- SDT.MIX.PD.start.par(n, data)
#-----------------------------------------------------------------
# Alternatively, one can use the number of data points instead of
# a data vector:
# par <- SDT.MIX.PD.start.par(n, 6)
#-----------------------------------------------------------------
data.frame(name = MIX.PD.par.names(par, n), par = par)
```

results in the following output (Symbols on the right were added by hand):

	name	par	
1	p.left[1]	0.00	($\pi_{1_{left}}$)
2	p.right[1]	0.00	($\pi_{1_{right}}$)
3	d'.left[1]	1.00	($d'_{1_{left}}$)
4	d'.right[1]	1.00	($d'_{1_{rightt}}$)
5	Mean[2]	0.00	(μ_2)
6	Stddev[2]	1.00	(σ_2)
7	p.left[2]	0.00	($\pi_{2_{left}}$)
8	p.right[2]	0.00	($\pi_{2_{right}}$)
9	d'.left[2]	1.00	($d'_{2_{left}}$)
10	d'.right[2]	1.00	($d'_{2_{rightt}}$)
11	t-1	-1.50	(t_1)
12	t-2	-0.75	(t_2)
13	t-3	0.00	(t_3)
14	t-4	0.75	(t_4)
15	t-5	1.50	(t_5)

Since the number of the signals is even (i.e. no foil signal is assumed to be present) the parameter vector does not comprise the mean μ_1 and standard deviation σ_1 for modeling the first pair (weak items). These are assumed to be fixed values: $\mu_1 = 0$ and $\sigma_1 = 1$.

(5) Order of data:

The data are response frequencies of different response categories for the different types of signals. The signals are assumed to occur in pairs (e.g. intact vs. rearranged items or items from two different sources, A and B). Additionally to the pairs the data of a foil signal may be present. The data corresponding to the foil signal must be presented first.

Comment: The output presents the data in the same order.

(6) Model function:

The function for computing model probabilities:

```
SDT.MIX.PD(par, n, fixed = NULL, ident = NULL)
```

`par` Vector of parameters

`n` Number of different types of signals (NOT types of pairs!).

`fixed` Matrix with fixed constraints. This matrix is required if fixed constraints on probability parameters are specified. This is due to the fact that the model computes the probability parameters form raw parameters (in vector `par`), whereas fixed probabilities are specified as probabilities in the range [0, 1].

`Ident` Not used within the function (present for compatibility reasons only)

Comment: The function can be used in isolation, e.g. for generating artificial data.

(7) Model matrix (analytical computation):

The model matrix (i.e., a $m \times p$ matrix of partial derivatives of the model probabilities with respect to the free parameters [m = number of generated probabilities (= number of data points), p = number of free model parameters] is computed by the following functions:

```
SDT.MIX.PD.Model.Matrix(par, n, fixed = NULL, ident = NULL, functional =
NULL)
```

`par` Vector of parameters

`n` Number of signal distributions (including the noise distribution)

`fixed` $2 \times q$ matrix of fixed constraints (cf. Chapter 4.1)

`ident` $2 \times r$ matrix of equality constraints (cf. Chapter 4.2)

`functional` Function implementing functional constraints. The function has to contain the attribute `"SDT.Jacobian"` containing a function for computing the Jacobian matrix of the functional constraints. If this function is not available in case of functional constraints being specified, the function computing the model matrix returns the value `NULL`. (For further details, see Chapter 4.4)

Comment: The function can be used in isolation, e.g. for specific tests of the model.

(8) Model matrix (numerical computation):

The following function computes the model matrix numerically, using the function `jacobian()` from the package `NumDeriv`.

```
SDT.MIX.PD.Matrix.Num(par, n, fixed = NULL, ident = NULL, functional =
NULL)
```

`par` Vector of parameters

`n` Number of signal distributions (including the noise distribution)

`fixed` $2 \times q$ matrix of fixed constraints (cf. Chapter 4.1)

`ident` $2 \times r$ matrix of equality constraints (cf. Chapter 4.2)

`functional` Function implementing functional constraints.

Comments: Contrary to the analytical counterpart this function does not require the specification of a function computing the Jacobian matrix in case of functional constraints being specified.

The function is also helpful for testing whether the analytical counterpart works properly.

The function is invoked for computing the model matrix in the function `SDT.Statistics` in the following situations:

(i) Functional constraints have been specified by the user without specifying a function for computing the Jacobian of the constraints

(ii) The flag `sym.gr` was set to `FALSE` in the estimation procedure thus preventing the computation of symbolic derivatives.

(9) Probability parameters (Free vs. fixed):
The probability parameters within the vector of starting parameters are raw parameters that can vary from -∞ to +∞. These parameters are transformed internally to probabilities by means of the logistic function:

$$\pi = \frac{\exp(p)}{1+\exp(p)},$$

where π denotes the computed probability and p symbolizes the raw parameter. Thus, a starting parameter of $p = 0$ corresponds to a probability of $\pi = 0.5$.

If a probability parameter is fixed by the user to a specific value (say 1.0), then the value provided is interpreted as a probability and not as a raw value. Thus, the program checks whether a parameter is fixed or not. In the first case, the parameter is interpreted as a fixed parameter whereas it is treated as a raw parameter that has to be transformed in the second case.

(10) Example: See Chapter 5.7.

3.4 The mixture model with two normal distributions per signal (MIX.2)

(1) Model structure:
Figure 4 depicts the structure of the model for a single type of signal.

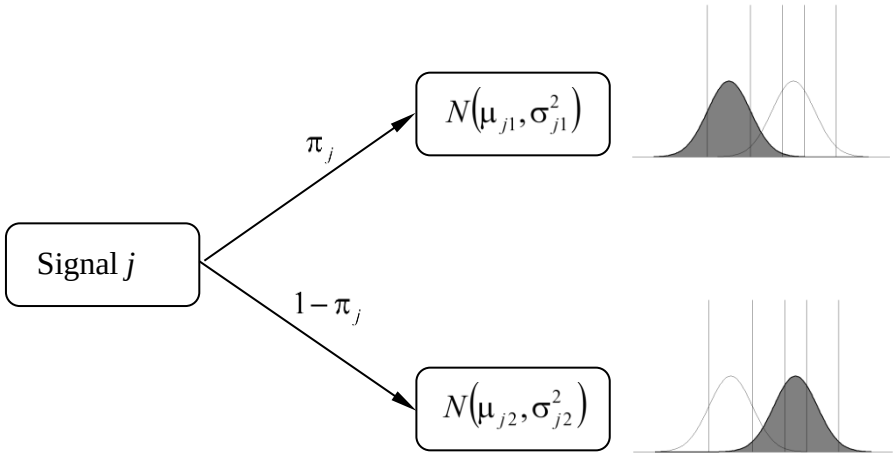


***Figure 4**: Structure of the MIX-2 model (mixture of two Gaussian distributions)*

The parameter π_j indicates the probability of the first distribution $N(\mu_{j1},\sigma^2_{j1})$, for each type of signal j $(j = 1,2,...,J)$.

For each type of signal a mixture of two distributions may be defined.

Comment: Without constraints on parameters the model is not identified.

(2) Model identification string: `"MIX.2"`

(3) Name of the file containing the model:
The source code of the model is contained in the file: `SDT-MIX2.R`.

(4) Order of parameters:

The parameters of the model are passed to the function `SDT.Estimate()` in the parameter `par`. For each type of signal *j* the parameters are given in the following order:

(i) π_j = Probability (in raw format [see below]) of the first Gaussian distribution for signal *j*.

(ii) μ_{j1} = Mean of the first Gaussian distribution for signal *j*.

(iii) σ_{j1} = Standard deviation of the first Gaussian distribution for signal *j*.

(iv) μ_{j2} = Mean of the second Gaussian distribution for signal *j*.

(v) σ_{j2} = Standard deviation of the second Gaussian distribution for signal *j*.

(vi) $t_{j1}, t_{j2}, \ldots, t_{j,R-1}$, = thresholds (decision bounds), where *R* is the number of response categories.

Help/Tip:

The function `SDT.MIX2.start.par(n, data)` creates starting parameters for the given number of signals `n` and the available data and the number of response categories per signal, respectively `data` (cf. Chapter 3.1, 3.2, or 3.3).
The function `MIX2.par.names(par, n)` creates names of the parameters in `par` for the number of signals `n` (that are also shown in the output).

(5) Order of data:

The data are response frequencies of the different response categories for the different types of signals.

Comment: The output presents the data in the same order.

(6) Model function:

The function for computing model probabilities:

```
SDT.MIX2(par, n, fixed = NULL, ident = NULL)
```

`par` Vector of parameters

`n` Number of types of signals (= number of mixtures).

`fixed` Matrix with fixed constraints. This matrix is required if fixed constraints on probability parameters are specified. This is due to the fact that the model computes the probability parameters form raw parameters (in vector `par`), whereas fixed probabilities are specified as probabilities in the range [0, 1].

`ident` Not used within the function (for compatibility reasons only)

Comment: The function can be used in isolation, e.g. for generating artificial data.

(7) Model matrix (analytical computation):

The model matrix (i.e., a $m \times p$ matrix of partial derivatives of the model probabilities with respect to the free parameters [m = number of generated probabilities (= number of data points), p = number of free model parameters] is computed by the following functions:

```
SDT.MIX2.Model.Matrix(par, n, fixed = NULL, ident = NULL, functional =
NULL)
```

`par` Vector of parameters

`n` Number of signal distributions (including the noise distribution)

`fixed` $2 \times q$ matrix of fixed constraints (cf. Chapter 4.1)

`ident` $2 \times r$ matrix of equality constraints (cf. Chapter 4.2)

`functional` Function implementing functional constraints. The function has to contain the attribute "`SDT.Jacobian`" containing a function for computing the Jacobian matrix of the functional constraints. If this function is not available in case of functional constraints being specified, the function computing the model matrix returns the value `NULL`. (For further details, see Chapter 4.4)

Comment: The function can be used in isolation, e.g. for specific tests of the model.

(8) Model matrix (numerical computation):

The following function computes the model matrix numerically, using the function `jacobian()` from the package `NumDeriv`.

```
SDT.MIX2.Matrix.Num(par, n, fixed = NULL, ident = NULL, functional =
NULL)
```

`par` Vector of parameters

`n` Number of signal distributions (including the noise distribution)

`fixed` $2 \times q$ matrix of fixed constraints (cf. Chapter 4.1)

`ident` $2 \times r$ matrix of equality constraints (cf. Chapter 4.2)

`functional` Function implementing functional constraints.

Comments: Contrary to the analytical counterpart this function does not require the specification of a function computing the Jacobian matrix in case of functional constraints being specified.

The function is also helpful for testing whether the analytical counterpart works properly.

The function is invoked for computing the model matrix in the function `SDT.Statistics` in the following situations:

(i) Functional constraints have been specified by the user without specifying a function for computing the Jacobian of the constraints

(ii) The flag `sym.gr` was set to `FALSE` in the estimation procedure thus preventing the computation of symbolic derivatives.

(9) Probability parameters (Free vs. fixed):

The probability parameters within the vector of starting parameters are raw parameters that can vary from $-\infty$ to $+\infty$. These parameters are transformed internally to probabilities by means of the logistic function:

$$\pi = \frac{\exp(p)}{1+\exp(p)},$$

where π denotes the computed probability and p symbolizes the raw parameter. Thus, a starting parameter of $p = 0$ corresponds to a probability of $\pi = 0.5$.

If a probability parameter is fixed by the user to a specific value (say 1.0), then the value provided is interpreted as a probability and not as a raw value. Thus, the program checks whether a parameter is fixed or not. In the first case, the parameter is interpreted as a fixed parameter whereas it is treated as a raw parameter that has to be transformed in the second case.

(10) Example:

See Chapter 7.

3.5 The dual process signal detection model (DPSDT)

The DPSDT model is a hybrid model consisting of a high-threshold component and a signal detection model. In memory research the threshold component may be interpreted as representing a discrete process of conscious recollection whereas the signal detection component may be interpreted as representing a continuous process based on familiarity.

Concerning the present implementation, the model contains a recollection probability parameter π representing the probability of selecting a specific response category, whereas with probability $1-\pi$ the signal detection component determines the selection of the response category (Figure 5). By default, the response category selected with probability π is the first response category for the first signal and the last response category for the other signals. However, is possible to change this assignment using the relevant option in the configuration list.

The DPSDT enables one to model source monitoring data (cf. Chapter 8.1) as well as data on associative recognition (cf. Chapter 8.2).

(1) Model structure:

For each signal *j* there is recollection component (represented by the recollection probability π_j) and a SDT component. Figure 5 depicts the structure of the model for two signals: Signal 1 and Signal *j* with the default assignment of the recollection probability to response categories (lowest category for the first signal and highest response category for the other signals).

The parameter π_1 represents the probability of choosing the lowest confidence category for the first signal. With probability of $(1-\pi_1)$ the probabilities of the response categories for the first signal are determined by the Gaussian distribution $N(\mu_1, \sigma_1^2)$.

The parameter π_j represents the probability of choosing the highest confidence category for the Signal j $(j = 2,...,J)$. With a probability of $(1-\pi_j)$ the probabilities of the response categories for Signal *j* are determined by the Gaussian distribution $N(\mu_j, \sigma_j^2)$.

(2) Model identification string: `"DPSDT"`

(3) Name of the file containing the model:

The source code of the model is contained in the file: `SDT-DPSDT.R`.

(4) Configuration information:

The configuration information is passed to the function `SDT.Estimate()` in the argument `n`. The configuration information consists of the following entries:

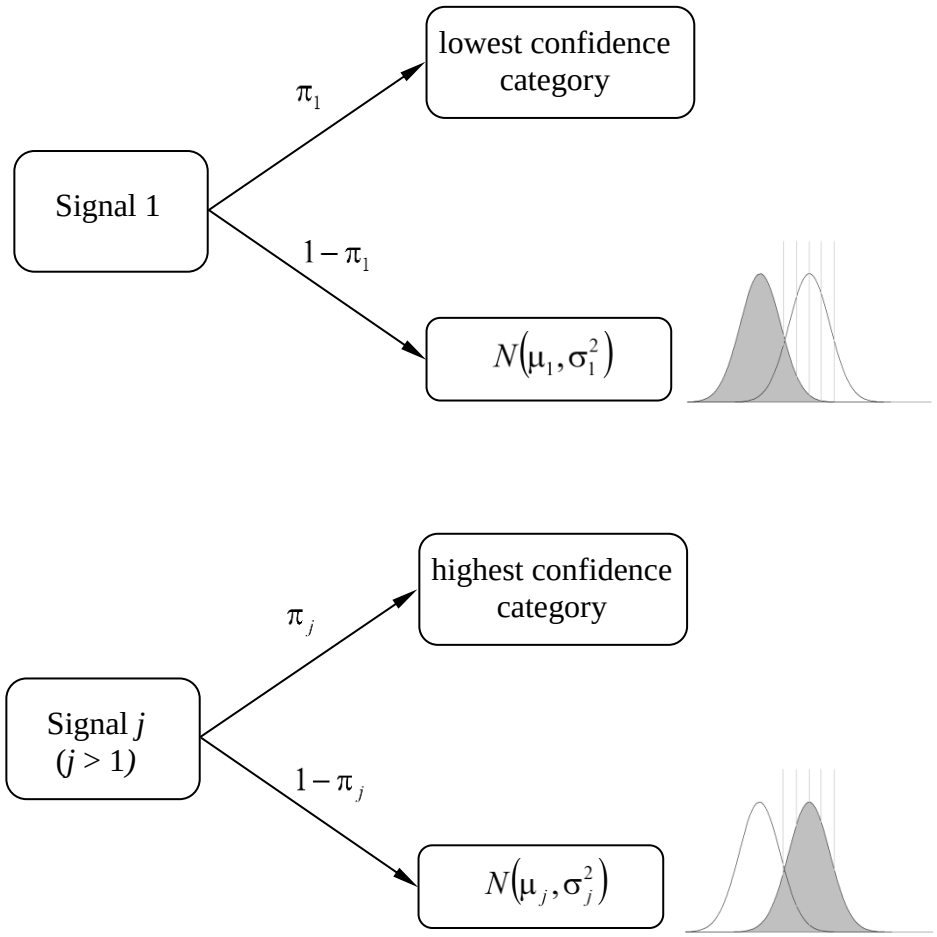


***Figure 5**: Structure of the DPSDT model (Dual process signal detection model).*

`n.sdt` = The number of signals (default: `n.sdt = 2`).

`rec.pos` = A vector representing the response categories that are selected in case of recollection. By default, for the first signal the first response category is selected and for the other signals the last response category is selected, for example with three signals and 6 response categories per signal the default looks like this: `rec.pos = c(1, 6, 6)`.

`restriction` = A string specifying the type of restrictions. The following options are available:

`"no"` No restrictions are specified (the model with this specification is not identified).

`"standard"`

The following restrictions are set:

1. The Gaussian distribution of the first signal is specified to be *N*(0, 1).
2. The variance parameters of the other distributions are set to $\sigma_j^2 = 1.0$.
3. The recollection parameter of the first signal is set to $\pi_1 = 0$.
4. The threshold parameters of all signals are set to be equal for all signals:
   $\tau_i^{\text{Signal1}} = \tau_i^{\text{Signal2}} = \cdots = \tau_i^{\text{SignalJ}} \quad (i = 1, \ldots R-1)$.

Comments:

- This set of restrictions is the default setting.
- The restrictions implement a single high-threshold model with signal detection component.

`"standard-2"`

Same as "`standard`", yet without the restriction on the recollection probability of the first model. Thus, contrary the standard setting the recollection parameter of the first signal π_1 is a free parameter.

The following restrictions are set:

1. The Gaussian distribution of the first signal is specified to be *N*(0, 1).
2. The variance parameters of the other distributions are set to $\sigma_j^2 = 1.0$.
3. The threshold parameters of all signals are set to be equal for all signals:
   $\tau_i^{\text{Signal1}} = \tau_i^{\text{Signal2}} = \cdots = \tau_i^{\text{SignalJ}} \quad (i = 1, \ldots R-1)$.

Comment:

The restrictions implement a double high-threshold model with signal detection component.

`"standard-2-eq"`

Same as "`standard-2`", yet with the additional restriction of recollection probabilities of all models being equal.

The following restrictions are set:

1. The Gaussian distribution of the first signal is specified to be *N*(0, 1).
2. The variance parameters of the other distributions are set to $\sigma_j^2 = 1.0$.
3. The recollection parameters of all models are assumed to be equal: $\pi_1 = \pi_2 = \cdots = \pi_J$
4. The threshold parameters of all signals are set to be equal for all signals:
   $\tau_i^{\text{Signal1}} = \tau_i^{\text{Signal2}} = \cdots = \tau_i^{\text{SignalJ}} \quad (i = 1, \ldots R-1)$.

`"standard-pairs"`

This option enables restriction for pairs of signals. For pairs of signals the HT-1 model is specified. Specifically, the following restrictions are set:

1. The mean of the Gaussian distribution for the first signal is set to zero $\mu_1 = 0$.
2. The recollection parameters of the first, third, fifth, etc. (uneven number of signals) are specified to be zero: $\pi_j = 0 \ (j = 1,3,5,...)$.
3. The variance parameters all distributions are set to $\sigma_j^2 = 1.0 \ (j = 1,...,J)$.
4. The threshold parameters of all signals are set to be equal for all signals:
   $\tau_i^{\text{Signal1}} = \tau_i^{\text{Signal2}} = \cdots = \tau_i^{\text{Signal}J} \ (i = 1,\ldots R-1)$.

Comment:
Works only with an equal number of signals.

`"standard-pairs-eq"`

Same as `standard-pairs` with the additional restriction of all variance parameter of pairs being the same. Specifically, the following restrictions are set:

1. The mean of the Gaussian distribution for the first signal is set to zero $\mu_1 = 0$.
2. The recollection parameters of the first, third, fifth, etc. (uneven number of signals) are specified to be zero: $\pi_j = 0 \ (j = 1,3,5,...)$.
3. The variance parameters for pairs of distributions are set to be equal:
   $\sigma_1^2 = \sigma_2^2 = 1.0$,
   $\sigma_3^2 = \sigma_4^2$,
   $\sigma_5^2 = \sigma_6^2$, etc.
4. The threshold parameters of all signals are set to be equal for all signals:
   $\tau_i^{\text{Signal1}} = \tau_i^{\text{Signal2}} = \cdots = \tau_i^{\text{Signal}J} \ (i = 1,\ldots R-1)$.

Comment:
Works only with an equal number of signals.

`"standard-lure-pairs"`

Same as `standard-pairs` with the additional assumption that the first distribution represents lures. Specifically, the following restrictions are set:

1. The first Gaussian distribution (representing lures is assumed to be $N(0, 1)$.
2. The recollection parameters of the second, fourth, sixth, etc. (even number of signals) are specified to be zero: $\pi_j = 0 \ (j = 2,4,6,...)$.

3. The variance parameters all distributions are set to $\sigma_j^2 = 1.0 \ (j = 1,...,J)$.
4. The threshold parameters of all signals are set to be equal for all signals:
   $\tau_i^{\text{Signal1}} = \tau_i^{\text{Signal2}} = \cdots = \tau_i^{\text{Signal}J} \ \ (i = 1,\ldots R-1)$.

Comment:
Works only with an unequal number of signals.

`"standard-lure-pairs-eq"`

Same as `standard-pairs-eq` with the additional assumption of the first distribution representing lures. Specifically, the following restrictions are set:

1. The first Gaussian distribution (representing lures is assumed to be *N*(0, 1).
2. The recollection parameters of the second, fourth, sixth etc. (even number of signals) are specified to be zero: $\pi_j = 0 \ (j = 2,4,6,...)$.
3. The variance parameters for pairs of distributions beginning from the second distribution are set to be equal:
   $\sigma_2^2 = \sigma_3^2$,
   $\sigma_4^2 = \sigma_5^2$, etc.
4. The threshold parameters of all signals are set to be equal for all signals:
   $\tau_i^{\text{Signal1}} = \tau_i^{\text{Signal2}} = \cdots = \tau_i^{\text{Signal}J} \ \ (i = 1,\ldots R-1)$.

Comment:
Works only with an unequal number of signals.

`"SDT"`

Restrictions conforming to the unequal variance signal detection model:

1. All recollection parameters are fixed to zero:
   $\pi_j = 0 \ \ (j = 1,...,J)$.
2. The Gaussian distribution of the first signal is specified to be *N*(0, 1).
3. The threshold parameters of all signals are set to be equal for all signals:
   $\tau_i^{\text{Signal1}} = \tau_i^{\text{Signal2}} = \cdots = \tau_i^{\text{Signal}J} \ \ (i = 1,\ldots R-1)$.

`"SDT-EV"`

Restrictions conforming to the equal variance signal detection model:

1. All recollection parameters are fixed to zero:
   $\pi_j = 0 \ \ (j = 1,...,J)$.
2. The Gaussian distribution of the first signal is specified to be *N*(0, 1).

3. The variance parameters of all models are set to $\sigma_j^2 = 1.0 \ \ (j = 1,...,J)$.
4. The threshold parameters of all signals are set to be equal for all signals:
   $\tau_i^{\text{Signal1}} = \tau_i^{\text{Signal2}} = \cdots = \tau_i^{\text{Signal}J} \ \ (i = 1,\ldots R-1)$.

Examples:
The file `DPSDT-3 (Parameter Information).R` demonstrates each of the constraints.

(5) Order of parameters:
The parameters of the model are passed to the function `SDT.Estimate()` in the parameter `par` in the following order:

For each of the *J* Gaussian models corresponding to the *J* signals $(j = 1,...,J)$ the following parameters have to specified in the following order:

(i) π_j = The probability of recollecting a specified response category in case of Signal *j*.

(ii) μ_j = Mean of the Gaussian distribution for Signal *j*.

(iii) σ_j = Standard deviation of the Gaussian distributions for Signal *j*.

(iv) $\tau_1, \tau_2, ..., \tau_{R-1}$ = thresholds (decision bounds), where *R* is the number of response categories for each Signal *j*. (identical for each signal).

Help/Tip:

The function `SDT.Parameter.Info()` can be used to display parameters as well as restrictions (cf. the example file: `DPSDT-3 (Parameter Information).R`).

(6) Order of data:

The data are response frequencies of different response categories for the different types of signals.

Comment: The output presents the data in the same order.

(7) Model function:
The function for computing model probabilities:

```
DPSDT.Probs(parvec, cfg, fixed = NULL, ident = NULL, with.p = T)
```

`parvec` Vector of parameters

`cfg` List with configuration information

`fixed` Matrix with fixed constraints. This matrix is required if fixed constraints on probability parameters are specified. This is due to the fact that the model computes the probability parameters form raw parameters (in vector `par`), whereas fixed probabilities are specified as probabilities in the range [0, 1].

`ident` Not used within the function (present for compatibility reasons only).

`with.p` A flag indicating whether probabilities should be computed from raw parameters (`with.p = T`).

Comment: The function can be used in isolation, e.g. for generating artificial data.

(8) Model matrix (analytical computation):
The model matrix (i.e., a *m*×*p* matrix of partial derivatives of the model probabilities with respect to the free parameters [*m* = number of generated probabilities (= number of data points), *p* = number of free model parameters] is computed by the following functions:

`DPSDT.Model.Matrix(full.par, cfg, fixed = NULL, ident = NULL, functional = NULL)`

`full.par`	Full parameter vector
`cfg`	List with configuration information
`fixed`	$2 \times q$ matrix of fixed constraints (cf. Chapter 4.1)
`ident`	$2 \times r$ matrix of equality constraints (cf. Chapter 4.2)
`functional`	Function implementing functional constraints. The function has to contain the attribute `"SDT.Jacobian"` containing a function for computing the Jacobian matrix of the functional constraints. If this function is not available in case of functional constraints being specified, the function computing the model matrix returns the value `NULL`. (For further details, see Chapter 4.4)

Comment: The function can be used in isolation, e.g. for specific tests of the model.

(9) Model matrix (numerical computation):

The following function computes the model matrix numerically, using the function `jacobian()` from the package `NumDeriv`.

`DPSDT.Matrix.Num(full.par, cfg, fixed = NULL, ident = NULL, functional = NULL)`

`full.par`	Full parameter vector
`cfg`	List with configuration information
`fixed`	$2 \times q$ matrix of fixed constraints (cf. Chapter 4.1)
`ident`	$2 \times r$ matrix of equality constraints (cf. Chapter 4.2)
`functional`	Function implementing functional constraints.

Comments: Contrary to the analytical counterpart this function does not require the specification of a function computing the Jacobian matrix in case of functional constraints being specified.

The function is also helpful for testing whether the analytical counterpart works properly.

The function is invoked for computing the model matrix in the function `SDT.Statistics` in the following situations:

(i) Functional constraints have been specified by the user without specifying a function for computing the Jacobian of the constraints

(ii) The flag `sym.gr` was set to `FALSE` in the estimation procedure thus preventing the computation of symbolic derivatives.

(10) Probability parameters (Free vs. fixed):

The probability parameters within the vector of starting parameters are raw parameters that can vary from -∞ to +∞. These parameters are transformed internally to probabilities by means of the logistic function:

$$\pi = \frac{\exp(p)}{1+\exp(p)},$$

where π denotes the computed probability and p symbolizes the raw parameter. Thus, a starting parameter of $p = 0$ corresponds to a probability of $\pi = 0.5$.

If a probability parameter is fixed by the user to a specific value (say 1.0), then the value provided is interpreted as a probability and not as a raw value. Thus, the program checks whether a parameter is fixed or not. In the first case, the parameter is interpreted as a fixed parameter whereas it is treated as a raw parameter that has to be transformed in the second case.

(11) Examples: See Chapter 8.

3.6 The double high-threshold model for modeling rating data (HT.n)

The HT.n model is a double high-threshold model with probabilities for modeling rating data. The model may be conceived of as a non-parametric counterpart to the MIX.2 model (cf. Chapter 3.4): Instead of using a mixture of parametric distributions a mixture of a Gaussian and a discrete distribution is used for modeling the data. The discrete distribution is represented by *n*-1 probability parameters with *n* denoting the number of response categories.
The HT.n model enables one to model complex recognition data (cf. Chapter 9.1 and Chapter 9.2). Similarly to the MIX.2 model, without the imposition of restriction the HT.n model is not identified.

(1) Model structure:
Figure 6 depicts the structure of the model for one signal.
The parameter π_j represents the mixing probability for Signal *j*, i.e., the probability that the discrete process represented by the discrete distribution is invoked.
With a probability of $(1-\pi_j)$ the probabilities of the response categories for the signal are determined by the Gaussian distribution $N(\mu_j, \sigma_j^2)$.

(2) Model identification string: `"HT.n"`

(3) Name of the file containing the model:
The source code of the model is contained in the file: `SDT-HT-n.R`.

(4) Order of parameters:
The parameters of the model are passed to the function `SDT.Estimate()` in the parameter `par` in the following order:
For each type of signal the following parameters have to provided in the following order:

(i) μ_j = Mean of the Gaussian distribution for Signal pair *j* (except for the first signal).

(ii) σ_j = Standard deviation of the Gaussian distributions for Signal *j* (except for the first signal).

(iii) π_j = Parameter representing the mixing probability, i.e., the probability of using the discrete distribution (instead of the Gaussian) for modeling the response distribution.

(iv) $\pi_j^1, \pi_j^2, ..., \pi_j^n$ = Probability parameters characterizing the discrete distribution (*n* = number of response categories).

(v) $t_1, t_2, ..., t_{n-1}$, = Threshold parameters (decision bounds), where *n* is the number of response categories per signal (assumed to be the same for each type of signal).

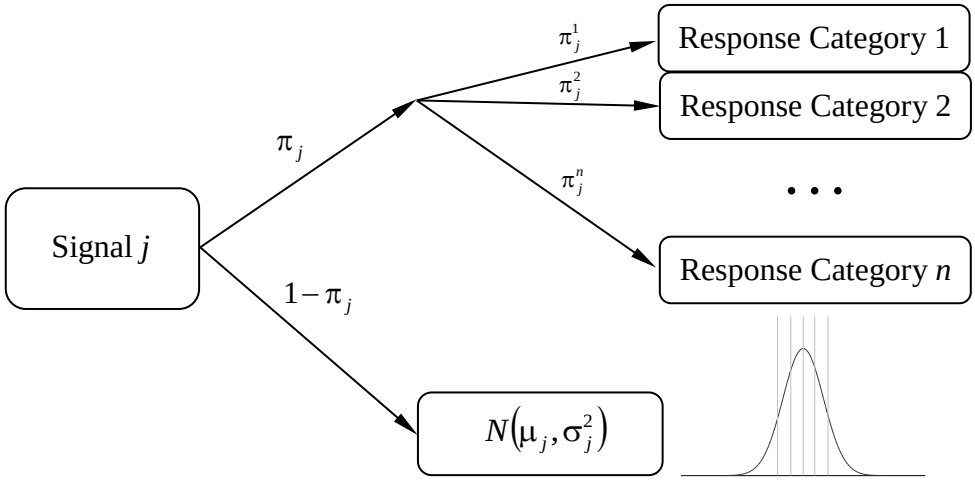


***Figure 6**: Structure of the HT.n model (Two high-threshold model for rating data).*

Comments:

- For the first signal no mean and variance parameter must be specified since these are supplied by the module: $\mu_1 = 0$ and $\sigma_1 = 1$.
- One of the probability parameters is redundant. This redundancy will be treated internally by the program. The redundant parameter must be included into the parameter vector to allow for the possibility of specifying a constraint on this parameter.

Help/Tip:

The function `SDT.HT.n.start.par(n, data, range.th = c(-1, 1))` creates starting parameters for the given number of signals `n` and the available data `data`. In case of `data` being a single number the value is interpreted as the number of response categories per signal.
The parameter `range.th` enables the specification of different ranges for the thresholds in the starting parameter. The first value indicates the value of the lowest and the second parameter indicates the value of the highest threshold.
The function `HT.n.par.names(par, n)` creates names of the parameters in `par` for the number of signals `n` (that are also shown in the output).

Example:

The sequence of commands:

```
par <- SDT.HT.n.start.par(2, 6)
data.frame(name = HT.n.par.names(par, n), par = par)
```

results in the following output (Symbols on the right were added by hand):

```
         name   par
1        p[1]  0.00      [π1]
2      p-1[1]  0.00      [π1^1]
3      p-2[1]  0.00      [π1^2]
4      p-3[1]  0.00      [π1^3]
```

```
5      p-4[1]  0.00      [π_1^4]
6      p-5[1]  0.00      [π_1^5]
7      p-6[1]  0.00      [π_1^6]
8     mean[2]  0.00      [μ_2]
9   stddev[2]  1.00      [σ_2]
10       p[2]  0.00      [π_2]
11     p-1[2]  0.00      [π_2^1]
12     p-2[2]  0.00      [π_2^2]
13     p-3[2]  0.00      [π_2^3]
14     p-4[2]  0.00      [π_2^4]
15     p-5[2]  0.00      [π_2^5]
16     p-6[2]  0.00      [π_2^6]
17        t-1 -1.00      [t_1]
18        t-2 -0.50      [t_2]
19        t-3  0.00      [t_3]
20        t-4  0.50      [t_4]
21        t-5  1.00      [t_5]
```

(5) Order of data:

The data are response frequencies of different response categories for the different types of signals.

Comment: The output presents the data in the same order.

(6) Model function:

The function for computing model probabilities:

```
SDT.HT.n(par, n, fixed = NULL, ident = NULL)
```

`par` Vector of parameters

`n` Number of different types of signals.

`Fixed` Matrix with fixed constraints. This matrix is required if fixed constraints on probability parameters are specified. This is due to the fact that the model computes the probability parameters form raw parameters (in vector `par`), whereas fixed probabilities are specified as probabilities in the range [0, 1].

`ident` Matrix with identity constraints

Comments:

1. The function can be used in isolation, e.g. for generating artificial data using raw parameters from which probability parameters are computed.
2. There exists an alternative function `SDT.HT.n.without(par, n, fixed = NULL, ident = NULL)` that assumes real probabilities as probability parameters instead of raw parameters. By consequence, no computation of probability from raw parameters ist performed.

(7) Model matrix (analytical computation):

The model matrix (i.e., a $m \times p$ matrix of partial derivatives of the model probabilities with respect to the free parameters [m = number of generated probabilities (= number of data points), p = number of free model parameters] is computed by the following functions:

```
SDT.HT.n.Model.Matrix(par, n, fixed = NULL, ident = NULL, functional =
NULL)
```

`par`	Vector of parameters
`n`	Number of signal distributions (including the noise distribution)
`fixed`	$2 \times q$ matrix of fixed constraints (cf. Chapter 4.1)
`ident`	$2 \times r$ matrix of equality constraints (cf. Chapter 4.2)
`functional`	Function implementing functional constraints. The function has to contain the attribute `" SDT.Jacobian"` containing a function for computing the Jacobian matrix of the functional constraints. If this function is not available in case of functional constraints being specified, the function computing the model matrix returns the value `NULL`. (For further details, see Chapter 4.4)

Comment: The function can be used in isolation, e.g. for specific tests of the model.

(8) Model matrix (numerical computation):

The following function computes the model matrix numerically, using the function `jacobian()` from the package `NumDeriv`.

```
SDT.HT.n.Model.Matrix.Num(par, n, fixed = NULL, ident = NULL, functional
= NULL)
```

`par`	Vector of parameters
`n`	Number of signal distributions (including the noise distribution)
`fixed`	$2 \times q$ matrix of fixed constraints (cf. Chapter 4.1)
`ident`	$2 \times r$ matrix of equality constraints (cf. Chapter 4.2)
`functional`	Function implementing functional constraints.

Comments: Contrary to the analytical counterpart this function does not require the specification of a function computing the Jacobian matrix in case of functional constraints being specified.

The function is also helpful for testing whether the analytical counterpart works properly.

The function is invoked for computing the model matrix in the function `SDT.Statistics` in the following situations:

(i) Functional constraints have been specified by the user without specifying a function for computing the Jacobian of the constraints

(ii) The flag `sym.gr` was set to `FALSE` in the estimation procedure thus preventing the computation of symbolic derivatives.

Comment: The function can be used in isolation, e.g. for specific tests of the model.

(9) Probability parameters (Free vs. fixed):
The probability parameters within the vector of starting parameters are raw parameters that can vary from -∞ to +∞. These parameters are transformed internally to probabilities by means of the logistic function:

$$\pi = \frac{\exp(p)}{1+\exp(p)},$$

where π denotes the computed probability and p symbolizes the raw parameter. Thus, a starting parameter of $p = 0$ corresponds to a probability of $\pi = 0.5$.

If a probability parameter is fixed by the user to a specific value (say 1.0), then the value provided is interpreted as a probability and not as a raw value. Thus, the program checks whether a parameter is fixed or not. In the first case, the parameter is interpreted as a fixed parameter whereas it is treated as a raw parameter that has to be transformed in the second case.

3.7 The bivariate Gaussian model of signal detection enabling violations of decisional separability on one dimension (SDT.2D)

The SDT.2D model enables the modeling of two dimensional detection data. The module contains the following features:

- ❒ Structural zeros may be specified;
- ❒ Violations of decisional separability on one dimension are allowed (cf. Ashby & Townsend, 1986);
- ❒ A matrix for pooling, projecting, and weighing etc. of the data and estimates can be specified. In this case a marginal likelihood estimation is performed (in case of pooling the data).
- ❒ An option for robust estimation of decision bounds is available. In case of this option being set the model decision bounds are implemented via positive increments thus ensuring the decision bounds to be ordered correctly. This option is required in case of estimating model using a pooling matrix (cf. Chapter 10.5).

(1) Model structure:
The model is described in detail in Wickens (1992) and in Wickens & Olzak (1992). Figure 17, Figure 18, and Figure 19 present graphical representations of the model with four signals.

(2) Model identification string: `"SDT.2D"`

(3) Name of the file containing the model:
The source code of the model is contained in the file: `SDT-SDT-2D.R`.

(4) Configuration information:
The argument `n` of the function `SDT.Estimate()` is a list containing information about the configuration of the model. The configuration list contains the following entries:

`n.sdt`: Number of signals (= number of Gaussian distributions);

Comment:
For compatibility purposes, instead of `n.sdt` the entry `s` can be used to indicate the number of signals.

`k1`: Number of decision bounds on Dimension 1;
`k2`: Number of decision bounds on Dimension 2;
`ds`: Flag indicating violations of decisional separability (default: `ds = 0`).

`ds = 0`: No violations of decisional separability;
`ds = 1`: Violations of decisional separability on Dimension 1;
`ds = 2`: Violations of decisional separability on Dimension 2;

`struct.zero`: Indices of those cells that are excluded from the process of estimation.

`R`: A response assignment matrix. This matrix enables the pooling of estimates from different cells. The data vector, as well as the vector of estimated probabilities is premultiplied by this matrix.

Comments:

- The data vector supplied has always to be of full length and the columns of matrix `R` must be equal to the length of the data vector.
- The employment of matrix R enables the estimation of bivariate Gaussian models with response selection (cf. DeCarlo, 2003b; Greene, 2008).
- For compatibility purposes, instead of `n.sdt` the entry `s` can be used to indicate the number of signals.

`conditional`: A flag indicating whether a conditional estimation should be performed. If the flag is set to `TRUE`, in case of structural zeros the residual probabilities for the respective signal are renormalized to sum to 1.0.

`restriction`: A character string indicating the types of restrictions to be specified. Two or more types of restrictions can be specified simultaneously. Currently, the following options are available:

`"st"`: This is the default option (`"st"` stands for (`"standard"`). In this case the following identification restrictions are set by the program: $\mu_{11} = \mu_{12} = 0.0$ and $\sigma_{11} = \sigma_{12} = 1.0$.
μ_{11} and μ_{12} denote the two means and σ_{11} and σ_{12} denote the standard deviations of the distribution of the first signal.

`"v1"` All variance parameters are set equal to 1.0

`"r0"` All correlations are set equal to 0.0

`"er"` All correlation parameters are restricted to be equal

`"qu"` rectangular configuration of Gaussians (only valid in case of four Gaussian models)

`"quvd"` rectangular configuration of Gaussians plus identical variances per dimension: $\sigma_{21} = \sigma_{11}$, $\sigma_{32} = \sigma_{12}$, $\sigma_{41} = \sigma_{31}$, $\sigma_{42} = \sigma_{22}$. (only valid in case of four Gaussian models). Figure 18 provides an illustration of this restriction.

`"no"`: No restriction is specified.

Comments:

- The restrictions are set additionally to those specified by the user. In case of conflicting restrictions concnering the same parameter, the user specified restrictions get precedence.
- If no standard restrictions are wanted the option `restriction = "no"` must be set (since `st` is the default option).
- It seems useful to separate the string indicating different types of restrictions (cf. the example below). However the program does not require this since it searches only for the respective string (ignoring cases).

`robust`: Flag indicating the robust estimation of decision bounds, i.e., the decision bounds are estimated as increments, in the following way:

$$\tau_1 = t_1,$$
$$\tau_2 = t_1 + \exp(t_2),$$
$$\tau_3 = t_1 + \exp(t_2) + \exp(t_3),$$
$$\ldots$$
$$\tau_k = t_1 + \exp(t_2) + \exp(t_3) + \cdots + \exp(t_{k-1}) + \exp(t_k),$$

where $t_1, t_2, \ldots, t_k$ are the raw parameters that are modified by the optimizer, whereas $\tau_1, \tau_2, \ldots, \tau_k$ denote the decision bounds computed internally and used for computing probabilities. This way of computing ensures the proper ordering of decision bounds,

Comments:

- The output presents the computed decision bounds. In addition the standard errors of the bounds are properly adjusted. By consequence the output is exactly identical independently of whether the flag was set.
- The option has however an influence on restrictions imposed on decision bounds. Restrictions concern raw parameters only and not computed ones. Thus, if, for example, two threshold parameter are set equal, the two decision bounds are not equal. Rather the increments corresponding increments are equal.
- For a concrete example of estimating a model using the `robust` option cf. Chapter 10.5.

Example:

```
n <- list(s = 4, k1 = 6, k2 = 6, ds = 1, robust = T, restriction = "st-v1-er-qu")
```

This list indicates the following setup for the SDT2D model:

- 4 types signals;
- 6 decision bounds on each dimension;
- Decisional separability on Dimension 1;
- Decision bounds are estimated robustly;
- The following types of restrictions are set by the program:
  - (i) Standard restrictions fixing the mean and standard deviation parameters of the first Gaussian distribution to 0.0 and 1.0 respectively.
  - (ii) The variance parameters of each Gaussian model are set equal to 1.0.
  - (iii) The correlation parameters of the four Gaussian models are assumed to be equal.
  - (iv) A rectangular arrangement of the Gaussian distributions is assumed.

  (A configuration of Gaussian models conforming to these restrictions is shown in Figure 17).

(5) Order of parameters:

The parameters of the model are passed to the function `SDT.Estimate()` in the parameter `par` in the following order:

A. For each type of signal (=Gaussian model) 5 parameters have to be provided in the following order:
   - (i) μ_1^s = First location parameter of the Gaussian distribution for Signal *s*;
   - (ii) μ_2^s = Second location parameter of the Gaussian distribution for Signal *s*;
   - (iii) σ_1^s = First standard deviation of the Gaussian distribution for Signal *s*;

(iv) σ_2^s = Second standard deviation of the Gaussian distribution for Signal *s*;

(v) ρ^s = Correlation parameter for the distribution representing Signal *s*.

B. The k_1 decision bounds on Dimension 1, in case of no violations of decisional separability (`ds = 0`) make up the second part of the parameter vector.
In case of violations of decisional separability on Dimension 1 (`ds = 1`), $k_1 \cdot (k_2 + 1)$ decision bounds are required, where k_2 is the number of decision bounds on Dimension 2.
The order of the decision bounds is illustrated in Figure 7. There are $k_1 = 3$ and $k_2 = 2$ decision bounds resulting in 6 response regions. $k_1 \cdot (k_2 + 1) = 9$ decision bounds have to be specified for Dimension 1. As shown in Figure 7, within each region on Dimension 2 the bounds for Dimension 1 are specified.

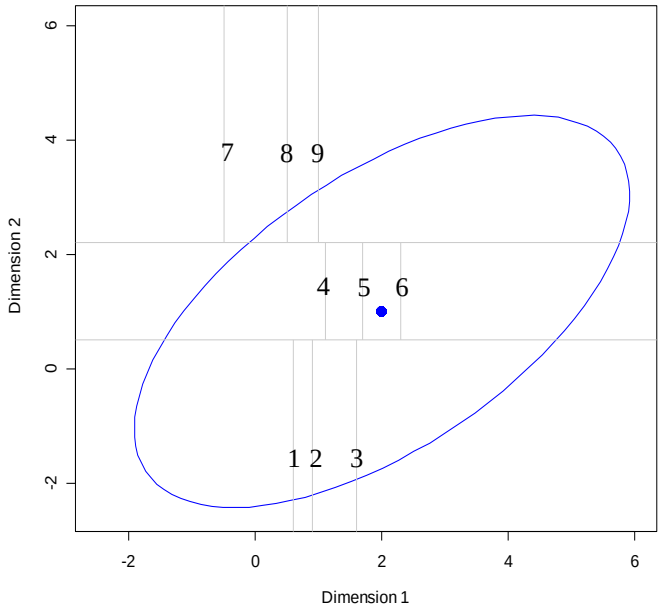


Figure 7: *SDT-2D model: Order of decision bounds in case of violations of decisional separability on Dimension 1.*

First the three bounds on Dimension 1 in the region $\left(-\infty, t_1^{D2}\right]$ of Dimension 2 are specified, where t_1^{D2} denotes the first decision bound on Dimension 2. Second the three bounds in the region $\left(t_1^{D2}, t_2^{D2}\right]$ of Dimension 2 are specified. Finally the three bounds in the regions $\left[t_2^{D2}, \infty\right)$ of Dimension 2 are specified.

C. The k_2 decision bounds on Dimension 2, in case of no violations of decisional separability (`ds = 0`) make up the third part of the parameter vector.
In case of violations of decisional separability on Dimension 2 (`ds = 2`), $k_2 \cdot (k_1 + 1)$ decision bounds are required.

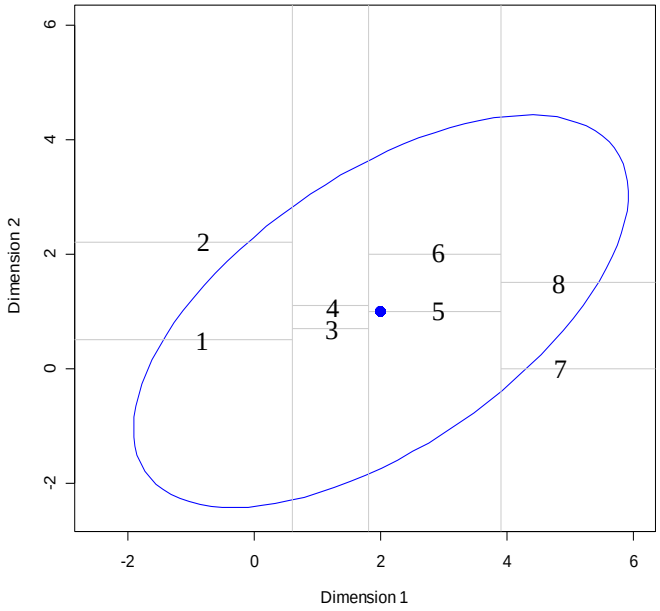


Figure 8: *SDT-2D model: Order of decision bounds in case of violations of decisional separability on Dimension 2.*

The order of the decision bounds is illustrated in Figure 8. Again, there are $k_1 = 3$ and $k_2 = 2$ decision bounds resulting in 6 response regions. $k_2 \cdot (k_1 + 1) = 8$ decision bounds have to be specified for Dimension 2. As shown in Figure 8, within each region on Dimension 1 the bounds for Dimension 2 are specified.

First, the two bounds on Dimension 1 in the region $\left(-\infty, t_1^{D1}\right]$ of Dimension 1 are specified, where t_1^{D1} denotes the first decision bound on Dimension 1. Second, the three bounds in the region $\left(t_1^{D1}, t_2^{D1}\right]$ of Dimension 1 are specified. Third, three bounds in the region $\left(t_2^{D1}, t_3^{D1}\right]$ of Dimension 1 are specified. Finally the three bounds in the regions $\left[t_3^{D1}, \infty\right)$ of Dimension 1 are specified.

Help/Tip:

The function `SDT.Parameter.Info(n, Model.Id = "SDT.2D")` provides the possibility for inspection of the model parameters, where the configuration list is passed to parameter `n`.

Example:

The order of parameters for the model in Figure 7 can be inspected by using of the following sequence of commands:

```
cfg <- list(s = 1, k1 = 3, k2 = 2, ds = 1)
Erg <- SDT.Parameter.Info(n = cfg, Model.Id = "SDT.2D")
print(Erg)
```

The output looks like this (symbols on the right were added):

```
$Parameters.and.Constraints
             name par fixed.value ident.source Nr
1       Mean.1[1]   0     0 <set>          ---        μ_1^1
```

2	Mean.2[1]	0	0 <set>	---	μ_2^1
3	Stddev.1[1]	1	1 <set>	---	σ_1^1
4	Stddev.2[1]	1	1 <set>	---	σ_2^1
5	Corr[1]	0	---	---	ρ^1
6	t.D1-1[D2 = 1]	-1	---	---	$t_{1,D_2=\left(-\infty,t_1^{D2}\right]}^{D_1}$
7	t.D1-2[D2 = 1]	0	---	---	$t_{2,D_2=\left(-\infty,t_1^{D2}\right]}^{D_1}$
8	t.D1-3[D2 = 1]	1	---	---	$t_{3,D_2=\left(-\infty,t_1^{D2}\right]}^{D_1}$
9	t.D1-1[D2 = 2]	-1	---	---	$t_{1,D_2=\left(t_1^{D2},t_2^{D2}\right]}^{D_1}$
10	t.D1-2[D2 = 2]	0	---	---	$t_{2,D_2=\left(t_1^{D2},t_2^{D2}\right]}^{D_1}$
11	t.D1-3[D2 = 2]	1	---	---	$t_{3,D_2=\left(t_1^{D2},t_2^{D2}\right]}^{D_1}$
12	t.D1-1[D2 = 3]	-1	---	---	$t_{1,D_2=\left[t_2^{D2},\infty\right)}^{D_1}$
13	t.D1-2[D2 = 3]	0	---	---	$t_{2,D_2=\left[t_2^{D2},\infty\right)}^{D_1}$
14	t.D1-3[D2 = 3]	1	---	---	$t_{3,D_2=\left[t_2^{D2},\infty\right)}^{D_1}$
15	t.D2-1	-1	---	---	$t_1^{D_2}$
16	t.D2-2	1	---	---	$t_2^{D_2}$

(6) Order of data:

The data are response frequencies of different response categories for the different types of signals.

Figure 9 illustrates the ordering of the input data assumed by the model. This ordering corresponds to the ordering of the estimated probabilities and frequencies provided by the model.

Figure 9 illustrates that the data »move faster« on Dimension 2 than on Dimension 1.

Comment: The output presents the data in the same order.

(7) Model function:

The function for computing model probabilities:

```
SDT.SDT2D(parvec, cfg, fixed = NULL, ident = NULL)
```

`parvec` The vector of parameters

`cfg` A list with information concerning the configuration of the model (for a detailed description of the entries of this list, see Chapter 2.1).

`fixed` Matrix with fixed constraints. This parameter is not used. It is provided for compatibility purpose only.

`Ident` Matrix with identity constraints. This parameter is not used. It is provided for compatibility purpose only.

Comment: The function can be used in isolation, e.g. for generating artificial data.

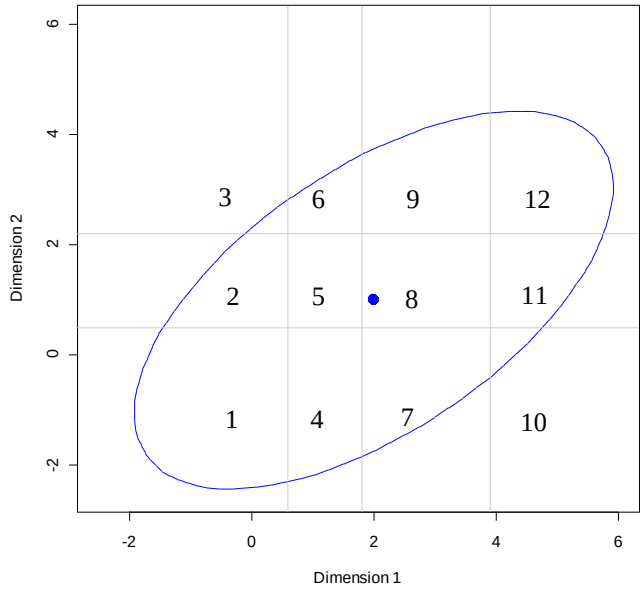


Figure 9: *SDT-2D model: Order of data and estimated probabilities and frequencies.*

(8) Model matrix (analytical computation):
The model matrix (i.e., a $m \times p$ matrix of partial derivatives of the model probabilities with respect to the free parameters [m = number of generated probabilities (= number of data points), p = number of free model parameters] is computed by the following functions:

```
SDT.SDT2D.Model.Matrix(full.par, cfg, fixed = NULL, ident = NULL,
functional = NULL)
```

`full.par`	Vector of parameters
`cfg`	A list with information about the configuration (for a detailed description of the entries of this list, see Chapter 2.1).
`fixed`	$2 \times q$ matrix of fixed constraints (cf. Chapter 4.1)
`ident`	$2 \times r$ matrix of equality constraints (cf. Chapter 4.2)
`functional`	Function implementing functional constraints. The function has to contain the attribute "`SDT.Jacobian`" containing a function for computing the Jacobian matrix of the functional constraints. If this function is not available in case of functional constraints being specified, the function computing the model matrix returns the value `NULL`. (For further details, see Chapter 4.4)

Comment: The function can be used in isolation, e.g. for specific tests of the model.

(9) Model matrix (numerical computation):
The following function computes the model matrix numerically, using the function `jacobian()` from the package `NumDeriv`.

```
SDT.SDT2D.Model.Matrix.Num(full.par, cfg, fixed = NULL, ident = NULL,
functional = NULL)
```

`par`	Vector of parameters
`cfg`	A list with information about the configuration (for a detailed description of the entries of this list, see Chapter 2.1).
`fixed`	$2 \times q$ matrix of fixed constraints (cf. Chapter 4.1)
`ident`	$2 \times r$ matrix of equality constraints (cf. Chapter 4.2)

`functional` Function implementing functional constraints.

Comments: Contrary to the analytical counterpart this function does not require the specification of a function for computing the Jacobian matrix in case of functional constraints being specified.
The function is also helpful for testing whether the analytical counterpart works properly.
The function is invoked for computing the model matrix in the function `SDT.Statistics` in the following situations:

(i) Functional constraints have been specified by the user without specifying a function for computing the Jacobian of the constraints

(ii) The flag `sym.gr` was set to `FALSE` in the estimation procedure thus preventing the computation of symbolic derivatives.

The function can be used in isolation, e.g. for specific tests of the model.

3.8 Mixture model of two bivariate Gaussian distributions per signal (SDT2D.MIX.2)

The SDT2D.MIX.2 model enables the modeling of two-dimensional detection data by means of a mixture of two bivariate Gaussian distributions per signal. The module contains the following features:

- ❒ Structural zeros may be specified;
- ❒ Violations of decisional separability on one dimension are allowed (cf. Ashby & Townsend, 1986);
- ❒ A matrix for pooling, projecting, and weighing etc. of the data and estimates can be specified. In this case a marginal likelihood estimation is performed (in case of pooling the data).
- ❒ An option for robust estimation of decision bounds is available. In case of this option being set the model decision bounds are implemented via positive increments thus ensuring the decision bounds to be ordered correctly. This option is required in case of estimating model using a pooling matrix (cf. Chapter 10.5). However, if this option is set, the usual specification of fixed and equality constraints does not work for decision bounds.

(1) Model structure:
Figure 10 depicts the structure of the model for a single signal (for a more complex example, cf. Figure 23 on p.194).

(2) Model identification string: `"SDT2D.MIX.2"`

(3) Name of the file containing the model:
The source code of the model is contained in the file: `SDT2D-MIX-2.R`.

(4) Configuration information:
The argument `n` of the function `SDT.Estimate()` is a list containing information about the configuration of the model. The configuration list contains the following entries:

`s`: Number of signals (= number of Gaussian distributions);
`k1`: Number of decision bounds on Dimension 1;
`k2`: Number of decision bounds on Dimension 2;
`ds`: Flag indicating violations of decisional separability (default: `ds = 0`).
`ds = 0`: No violations of decisional separability;
`ds = 1`: Violations of decisional separability on Dimension 1;
`ds = 2`: Violations of decisional separability on Dimension 2;
`Struct.zero`: Indices of those cells that are excluded from the process of estimation.

`R`: A response assignment matrix. This matrix enables the pooling of estimates from different cells. The data vector, as well as the vector of estimated probabilities is premultiplied by this matrix.

Comments:

- The data vector supplied has always to be of full length and the columns of matrix `R` must be equal to the length of the data vector.
- The employment of matrix R enables the estimation of bivariate Gaussian models with response selection (cf. DeCarlo, 2003b; Greene, 2008).

`conditional`: A flag indicating whether a conditional estimation should be performed. If the flag is set to `TRUE`, in case of structural zeros the residual probabilities for the respective signal are renormalized to sum to 1.0.

`standard`: Flag indicating whether the standard restrictions for fixing the location and scale of the configuration is used: $\mu_{11} = \mu_{12} = 0$ and $\sigma_{11} = \sigma_{12} = 1$,
where μ_{11} and μ_{12} denote the two means and σ_{11} and σ_{12} denote the standard deviations of the distribution of the first signal.
S`tandard = TRUE`: The restrictions are set by the program. Thus they need not be specified by the user (This is the default option if no value for `standard` was specified);
s`tandard = FALSE`: The constraints necessary for identification have to be specified by the user.

Comment:
If the restrictions have been specified by the user the option `standard` is ignored (i.e., the specification of the user is employed).

`Robust`: Flag indicating the robust estimation of decision bounds, i.e., the decision bounds are estimated as increments, in the following way:

$$\tau_1 = t_1,$$
$$\tau_2 = t_1 + \exp(t_2),$$
$$\tau_3 = t_1 + \exp(t_2) + \exp(t_3),$$
...
$$\tau_k = t_1 + \exp(t_2) + \exp(t_3) + \cdots + \exp(t_{k-1}) + \exp(t_k),$$

where $t_1, t_2, \ldots, t_k$ are the raw parameters that are modified by the optimizer, whereas $\tau_1, \tau_2, \ldots, \tau_k$ denote the decision bounds computed internally and used for computing probabilities. This way of computing ensures the proper ordering of decision bounds,

Comments:

- The output presents the computed decision bounds. In addition the standard errors of the bounds are properly adjusted. By consequence the output is exactly identical independently of whether the flag was set.
- The option has however an influence on restrictions imposed on decision bounds. Restrictions concern raw parameters only and not computed ones. Thus, if, for example, two threshold parameter are set equal, the two decision bounds are not equal. Rather the increments corresponding increments are equal.
- For a concrete example of estimating a model using the `robust` option cf. Chapter 10.5.

Example:

```
n <- list(s = 4, k1 = 6, k2 = 6, ds = 1, robust = T)
```

This list indicates the following setup for the SDT2D-MIX-2 model:

(11) 4 types signals;
(12) 6 decision bounds on each dimension;
(13) Decisional separability on Dimension 1;
(14) Decision bounds are estimated robustly;
(15) Standard restrictions are used (due to a missing specification, `standard = T`).

(5) Order of parameters:

The parameters of the model are passed to the function `SDT.Estimate()` in the parameter `par` in the following order:

A. For each type of signal (=Gaussian model) 11 parameters have to be provided in the following order:
   (i) π^s = Probability weight for the first of the two bivariate distributions
   (ii) $\mu^s_{x_1}$ = Location parameter on *x*-dimension of the first Gaussian distribution for Signal *s*;
   (iii) $\mu^s_{y_1}$ = Location parameter on *y*-Dimension of the first Gaussian distribution for Signal *s*;
   (iv) $\sigma^s_{x_1}$ = Scale parameter on *x*-dimension of the first Gaussian distribution for Signal *s*;
   (v) $\sigma^s_{y_1}$ = Scale parameter on *y*-dimension of the first Gaussian distribution for Signal *s*;
   (vi) ρ^s_1 = Correlation parameter for the first Gaussian distribution for Signal *s*.
   (vii) $\mu^s_{x_2}$ = Location parameter on *x*-dimension of the second Gaussian distribution for Signal *s*;
   (viii) $\mu^s_{y_2}$ = Location parameter on *y*-Dimension of the second Gaussian distribution for Signal *s*;
   (ix) $\sigma^s_{x_2}$ = Scale parameter on *x*-dimension of the second Gaussian distribution for Signal *s*;
   (x) $\sigma^s_{y_2}$ = Scale parameter on *y*-dimension of the second Gaussian distribution for Signal *s*;
   (xi) ρ^s_2 = Correlation parameter for the second Gaussian distribution for Signal *s*.

B. The k_1 decision bounds on Dimension 1, in case of no violations of decisional separability (`ds = 0`) make up the second part of the parameter vector.

   In case of violations of decisional separability on Dimension 1 (`ds = 1`), $k_1 \cdot (k_2 + 1)$ decision bounds are required, where k_2 is the number of decision bounds on Dimension 2.

   The order of the decision bounds is illustrated in Figure 7. There are $k_1 = 3$ and $k_2 = 2$ decision bounds resulting in 6 response regions. $k_1 \cdot (k_2 + 1) = 9$ decision bounds have to be specified for Dimension 1. As shown in Figure 7, within each region on Dimension 2 the bounds for Dimension 1 are specified.

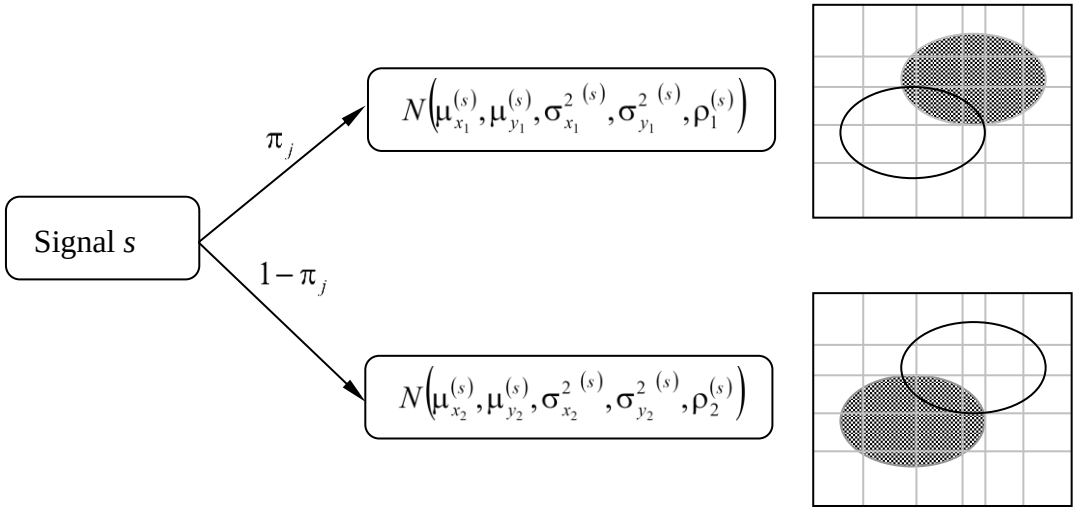


***Figure 10**: Structure of the SDT-MIX-2 model (mixture of two bivariate Gaussian distributions).*

Example:
Given the following configuration:

- 1 signal;
- 3 decision bounds on Dimension 1 (*x*-dimension);
- 2 decision bounds on Dimension 2 (*y*-dimension);
- No violation of decision separability.

The following sequence of commands:

```
cfg <- list(s = 1, k1 = 3, k2 = 2, ds = 0)
Erg <- SDT.Parameter.Info(n = cfg, Model.Id = "SDT2D.MIX.2")
print(Erg)
```

provides an output of the ordering (as well as of the starting values) of the parameters:

```
$Model
[1] "Bivariate Gaussian 2 Mixture model: No violations of decisional
separability [STANDARD RESTRICTIONS]"

$Parameters.and.Constraints
            name  par fixed.value ident.source Nr
1           p[1]  0.0         ---          ---
2    Mean.x[1.1]  0.0     <set> 0          ---
3    Mean.y[1.1]  0.0     <set> 0          ---
4  Stddev.x[1.1]  1.0     <set> 1          ---
5  Stddev.y[1.1]  1.0     <set> 1          ---
6      Corr[1.1]  0.0         ---          ---
7    Mean.x[1.2] -0.1         ---          ---
8    Mean.y[1.2] -0.1         ---          ---
9  Stddev.x[1.2]  1.0         ---          ---
10 Stddev.y[1.2]  1.0         ---          ---
11     Corr[1.2]  0.0         ---          ---
12        t.D1-1 -1.0         ---          ---
13        t.D1-2  0.0         ---          ---
14        t.D1-3  1.0         ---          ---
15        t.D2-1 -1.0         ---          ---
16        t.D2-2  1.0         ---          ---
```

Comments:

1. The first number of the in the square brackets indicates the number of the model, and the second one (if present) indicates the Gaussian distribution. For instance, `Mean.x[1.2]` is the mean parameter on the *x*-dimension of the second Gaussian distribution for Signal 1.
2. The symbol `<set> 0` etc. indicates that the parameter is set by default to the values 0, and similar for the other parameters. These constraints are provided by the program in order to make the model identifiable. The provision of standard constraints by the program can suppressed by specifying the option `standard = F` in the configuration list. In this case, the constraints for making the model identifiable have to provided by the user.
3. For further information on the function `SDT.Parameter.Info()`, cf. Chapter 2.4.

(6) Order of the input data:
The order of the input data and of the estimated frequencies is identical to that of the SDT-2D model (cf. Figure 9).

3.9 Gaussian SDT model for *m*-alternative forced choice data (mAFC)

The mAFC model enables the modeling of *m*-alternative forced choice data with bias, assuming that the signal distribution of the *m* alternatives conform to Gaussian distributions.

(1) Model structure:
The model is described in DeCarlo (2012). The implemented model is slightly more general by enabling the estimation of a variance parameter for the signal distribution of the target alternative.

(2) Model identification string: `"mAFC"`

(3) Name of the file containing the model:
The source code of the model is contained in the file: `SDT-mAFC.R`.

(4) Configuration information:
The argument `n` of the function `SDT.Estimate()` is a list containing information about the configuration of the model. The list contains the following entries:

`s`: Number of conditions (default: `s = 1`).
`m`: Number of possible alternatives (the specification of this entry is obligatory).
`npt`: Number of quadrature points (default: `npt = 25`).
`quad`: Method of quadrature: `"aGH"` = adaptive Gauss-Hermite or `"GH"` = Gauss-Hermite quadrature (default: `quad = "GH"`).

Example:

```
n <- list(s = 2, m = 3, npt = 35, quad = "GH")
```

This list indicates the following setup for the `mAFC` model:

(16) 2 experimental conditions;
(17) 3 possible alternatives;
(18) 35 quadrature points;
(19) Gauss-Hermite quadrature.

Alternatively, `n` may be a single integer that specifies the number of alternatives in the set of possible alternatives.

(5) Order of parameters:
The parameters of the model are passed to the function `SDT.Estimate()` in the parameter `par` in the following order:

For each experimental conditions the following $m + 1$ parameters have to be specified in the following order:

(i) μ_S = Mean of the target signal distribution (i.e., the alternative with the signal). The means of the other distributions are assumed to be zero.

(ii) σ_S = Variance parameter of the target signal distribution. The variances of the other distributions are assumed to be one.

(iii) $\beta_1,\ldots,\beta_m$ = $m-1$ bias parameters for the intervals 1 to m. The bias parameter of the last interval is assumed to be zero.

Example:
Given the following configuration:

- 2 experimental conditions,
- 4 alternatives,

The following sequence of commands:

```
cfg <- list(s = 2, m = 4)
Erg <- SDT.Parameter.Info(n = cfg, Model.Id = "mAFC")
print(Erg)
```

provides an output of the ordering (as well as of the starting values) of the parameters:

```
$Model
[1] "4-AFC Model with Bias, Assuming Gaussian Distributions, Number of
Models: 2"

$Parameters.and.Constraints
    name par fixed.value ident.source Nr
1  d'[1] 0.5         ---          ---
2  sd[1] 1.0         ---          ---
3  b1[1] 0.0         ---          ---
4  b2[1] 0.0         ---          ---
5  b3[1] 0.0         ---          ---
6  d'[2] 0.5         ---          ---
7  sd[2] 1.0         ---          ---
8  b1[2] 0.0         ---          ---
9  b2[2] 0.0         ---          ---
10 b3[2] 0.0         ---          ---
```

Comments:

1. `d'` denotes μ_S, the mean of the target signal distribution;
2. `sd` denotes σ_S, the variance of the target signal distribution;
3. `b1` denotes the bias β_1 of the first interval (the alternative presented first);
4. `b2` denotes the bias β_2 of the second interval (the alternative presented first);
5. `b3` denotes the bias β_3 of the third interval (the alternative presented first);
6. The numbers in square brackets denote the experimental conditions.

(7) Order of the input data:
For each experimental condition a vector of $m \cdot m$ data have to be specified: in the following order:

- Choice frequencies of the *m* alternatives with the target alternative in the first interval.
- Choice frequencies of the *m* alternatives with the target alternative in the second interval.

  …
- Choice frequencies of the *m* alternatives with the target alternative in the *m*-th interval.

Example: Cf. Chapter 12.

3.10 Gaussian SDT model for *k*-alternative forced choice data with and without rating data (SDT.Rank)

The SDT-Rank model enables the simultaneous modeling of multiple sets of *k*-alternative forced choice data alone or together with multiple sets of rating data. The different sets of forced choice data may comprise forced choices with different numbers of distractors. The set of reating models can be used to model rating data with different response categories. The model implements actually two different types of repeated AFC models, a parallel and a serial one.

(1) Model structure of the AFC models:

A. Parallel version:

The probability of choosing the target alternative out of *k* alternatives in trial *j* (or alternatively the probability that the target alternative has rank j $(j = 1,2,\ldots,k)$ is given by the equation (cf. Equation 6 on p. 468 of Kellen, Klauer, & Singmann, 2012):

$$\pi_j = \int_{-\infty}^{\infty} \binom{k-1}{j-1} \cdot \left[1 - \Phi\left(x|\mu_N,\sigma_N\right)\right]^{j-1} \left[\Phi\left(x|\mu_N,\sigma_N\right)\right]^{k-j} \cdot \phi\left(x|\mu_S,\sigma_S\right) dx$$

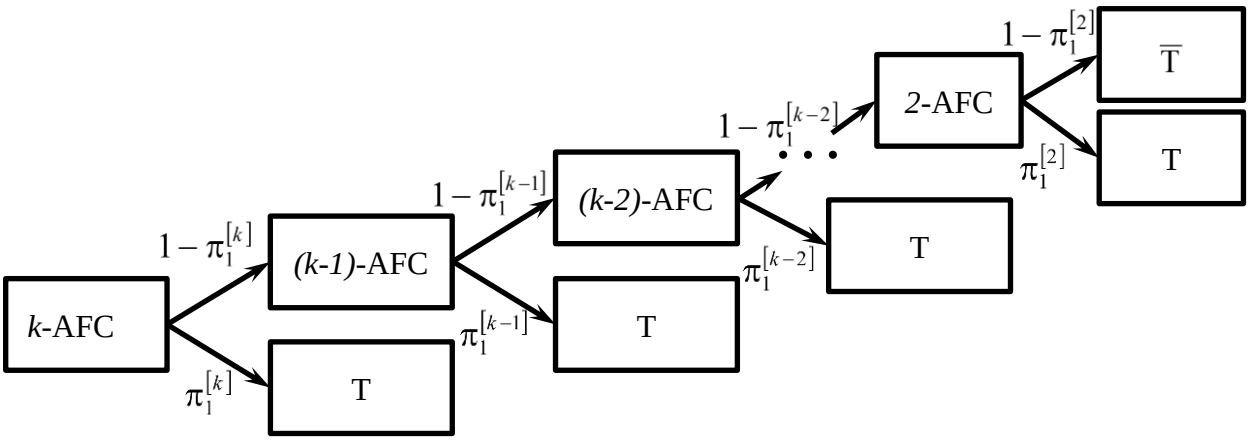
where:

π_j	denotes the probability that the old stimulus (or target stimulus) is in position *j*;
k	represents the number of alternatives presented;
$\Phi\left(x\|\mu_N,\sigma_N\right)$	denotes the cumulative Gaussian distribution with mean μ_N and standard deviation σ_N (using standard restrictions: $\mu_N = 0$ and $\sigma_N = 1$);
$\phi\left(x\|\mu_S,\sigma_S\right)$	symbolizes the Gaussian density with mean μ_S and standard deviation σ_S.

B. Serial version:

The serial version assumes a task with repeated forced choices where after each choice of a non-target the erroneously selected item is eliminated from the set of items. For example, assuming 4 alternatives (the target and three distractors), all four items are presented at the beginning. If a non-target item is selected this item is removed reducing the task to a 3-AFC task (the target item and two distractors). As soon as the target item is selected, the task ends. The whole task may thus be conceived of as a series of AFC task with a reduction of the number of alternatives in each step (if a non-target is selected).

The associated model can thus be represented by means of a processing tree:



where:

k denotes the number of alternatives at the beginning of the selection;

$\pi_1^{[k]}$ denotes the probability that the old stimulus (or target stimulus T) is selected;

T the target item was selected;

$\overline{\text{T}}$ the non-target item was selected.

(2) Model structure of the Rating parts:
The model for modeling the rating data corresponds to the unequal variance signal detection (UVSDT) model (cf. Chapter 3.1).

(3) Model identification string: `"SDT.Rank"`

(4) Name of the file containing the model:
The source code of the model is contained in the file: `SDT-Rank.R`.

(5) Configuration information:
The argument `n` of the function `SDT.Estimate()` is a list containing information about the configuration of the model. The list contains the following entries:

`n.sdt` A 2×*k* matrix indicating the number of Gaussians and the number of response categories for the *k* rating models:
- ❒ The first row indicates the number of Gaussian distributions of the rating model.
- ❒ The second row indicates the number of response categories.

Comment: It is possible to simply provide a vector as an argument, e.g.
`n.sdt = c(2, 6, 2, 4).`
This is transformed to a matrix with the first row containing the entries [2, 2] and the second row containing the entries [6, 4]. This tells the program that there are two different types of rating data:
(a) Rating data with two types of stimuli and 6 response categories, and
(b) Rating data with two types of stimuli and 4 response categories.

`model` A vector of strings of `"P"` and `"S"` indicating the ranking models that are used for the different data sets. `"P"` indicates a parallel model and `"S"` denotes a serial model (default: `model="P"`).

`k.rg`: A vector with the number of alternatives for the different data sets.

`j.rg`: A vector with the number of relevant rank positions that are computed for the different ranking models. The numbers may be integers between 2 and `k.rg`. The default value is `k.rg`.

`npt`: Number of quadrature points for Gauss-Hermite quadrature (default: `npt = 30`).

`restriction`: A string indicating the type of restrictions to be set. There are two possibilities:

`standard`: standard restrictions of identification constraints (see below).

`equal`: standard restrictions plus equality restrictions on means and standard deviations fopr all Gaussians (ranking & rating).

`no`: No restrictions.

(default `restriction = standard`).

`rating`: A flag indicating whether only ranking data are to be modeled (`rating = F`) or ranking as well as rating data (default: `rating = T`)

Example 1:

```
cfg <- list(n.sdt = c(2, 6, 2, 4), model = c("P", "S"), k.rg = c(4, 3),
rating = T, restriction = "equal")
```

This list indicates the following setup for the `SDT.Rank` model:

1. There are two rating models with 2 Gaussian distributions each. For the first model there are 6 response categories and for the second model there are 4 response categories;
2. Two different ranking models for the two data sets are used: For the first set a parallel model is applied and for the second one a serial model is applied;
3. Two ranking data (or repated AFC data) sets with 4 and 3 alternatives;
4. Ranking and rating data are modeled.
5. Standard + Equality restrictions are set.

(6) Order of parameters:

The parameters of the model are passed to the function `SDT.Estimate()` in the parameter `par` in the following order (if no parameter vector is provided, the model generates a vector of starting parameters):

A. For each forced choice model four parameters have to be specified:
   (i) μ_N = Mean parameter of the new distribution (non-target distribution).
   (ii) σ_N = Variance parameter of the new distribution (non-target).
   (iii) μ_S = Mean parameter of the old distribution (target distribution).
   (iv) σ_S = Variance parameter of the old distribution (target distribution).

B. For each rating model of the SDT component the following parameters have to be specified (for each Gaussian):
   (i) μ = Mean parameter of the Gaussian distribution (for each Gaussian of the model).
   (ii) σ = Variance parameter of the old distribution (for each Gaussian in the model).

C. After the specification of the Gaussian parameters for each Gaussian of the rating model, the threshold parameters of the rating model (in case of rating data being modeled) have to be specified:
   $\tau_1,\ldots,\tau_m$ threshold parameters.

The paraemters in B and C are repeated for each rating model.
If `rating = F`, i.e. no rating data are modeled, only parameters in A have to be specified.

Example 1:
The following sequence of commands:

```
cfg <- list(n.sdt = c(2, 6, 2, 4), model = c("P", "S"), k.rg = c(4,
3), rating = T, restriction = "equal"))
PI <- SDT.Parameter.Info(n = cfg, Model.Id = "SDT.Rank")
print(PI)
```

provides the following output:

```
$Model
[1] "SDT ranking and rating model: Number of Gaussians within SDT
models: <<2,6>, <2,4>> Model types: <P,S> Number of ranking-
alternatives: <4,3> Number of ranking-positions: <4,3> Number of
quadrature points: <30> Restrictions: <equal>"
$Parameters.and.Constraints
                name   par fixed.value         ident.source Nr
1    [P-1]   Mean.1  0.00     0 <set>                  ---
2    [P-1] Stddev.1  1.00     1 <set>                  ---
3    [P-1]   Mean.2  0.50        ---                   ---
4    [P-1] Stddev.2  1.00        ---                   ---
5    [S-2]   Mean.1  0.00     0 <set>                  ---
6    [S-2] Stddev.1  1.00     1 <set>                  ---
7    [S-2]   Mean.2  0.50        --- [P-1]   Mean.2 <set>  3
8    [S-2] Stddev.2  1.00        --- [P-1] Stddev.2 <set>  4
9  [SDT-1]   Mean.1  0.00     0 <set>                  ---
10 [SDT-1] Stddev.1  1.00     1 <set>                  ---
11 [SDT-1]   Mean.2  0.00        --- [P-1]   Mean.2 <set>  3
12 [SDT-1] Stddev.2  1.00        --- [P-1] Stddev.2 <set>  4
13 [SDT-1]      c-1 -0.50        ---                   ---
14 [SDT-1]      c-2 -0.25        ---                   ---
15 [SDT-1]      c-3  0.00        ---                   ---
16 [SDT-1]      c-4  0.25        ---                   ---
17 [SDT-1]      c-5  0.50        ---                   ---
18 [SDT-2]   Mean.1  0.00     0 <set>                  ---
19 [SDT-2] Stddev.1  1.00     1 <set>                  ---
20 [SDT-2]   Mean.2  0.00        --- [P-1]   Mean.2 <set>  3
21 [SDT-2] Stddev.2  1.00        --- [P-1] Stddev.2 <set>  4
22 [SDT-2]      c-1 -0.50        ---                   ---
23 [SDT-2]      c-2  0.00        ---                   ---
24 [SDT-2]      c-3  0.50        ---                   ---
```

Comments:

1. `[P-1]` indicates that the first ranking model is a parallel model;
   `[S-2]` indicates that the second ranking model is a serial model.
2. `Mean.1` denotes μ_N, the mean of the new (non-target) distribution for each model;
3. `Stddev.1` denotes σ_N, the variance of the new (non-target) distribution for each model;
4. `Mean.2` denotes μ_S, the mean of the signal (target) distribution for each model;
5. `Stddev.2` denotes σ_S, the variance of the old (target) distribution for each model.

6. Due to the fact that `restriction = "equal"` the following restrictions were set:
   (i) The mean and the standard deviation of the first Gaussian distribution (representing non-targets) are set to 0 and 1.
   (ii) The mean and standard deviation of the Gaussians (representing targets) in the second model (as well as of all further models are set equal to the parameter of the Gaussian of the target distribution of first AFC model.
7. The values in column `par` are the values of the starting parameters (for the estimateon procedure).

Example 2:
The following sequence of commands:

```
cfg <- list(n.sdt = c(2, 6), model = c("S", "P", "S"), k.rg = c(4, 3,
4), rating = T)
P <- SDT.Parameter.Info(n = cfg, Model.Id = "SDT.Rank")
```

provides an output of the ordering (as well as of the starting values) of the parameters:

```
$Model
[1] "SDT ranking model: Model types: <S,P,S> Number of ranking-
alternatives: <4,3,4> Number of ranking-positions: <4,3,4> Number of
quadrature points: <30> Restrictions: <standard>"
$Parameters.and.Constraints
              name par fixed.value ident.source Nr
1  [S-1]   Mean.1 0.0     0 <set>          ---
2  [S-1] Stddev.1 1.0     1 <set>          ---
3  [S-1]   Mean.2 0.5         ---          ---
4  [S-1] Stddev.2 1.0         ---          ---
5  [P-2]   Mean.1 0.0     0 <set>          ---
6  [P-2] Stddev.1 1.0     1 <set>          ---
7  [P-2]   Mean.2 0.5         ---          ---
8  [P-2] Stddev.2 1.0         ---          ---
9  [S-3]   Mean.1 0.0     0 <set>          ---
10 [S-3] Stddev.1 1.0     1 <set>          ---
11 [S-3]   Mean.2 0.5         ---          ---
12 [S-3] Stddev.2 1.0         ---          ---
```

Comments:
1. Now there are three sets of ranking data:
   `[S-1]` indicates that the first model (for modeling the first data set) is a parallel model;
   `[P-2]` indicates that the second model (for modeling the second data set) is a serial model;
   `[S-3]` indicates that the third model (for modeling the third data set) is a serial model.
2. Due to the fact that `restriction = standard` (by default) the following restrictions were set:
   The mean and the standard deviation of the first Gaussian distribution (representing non-targets) are set to 0 and 1 for each model.
3. Due to the fact that only the ranking data are modeled (`rating = F`), no parameters for the rating model are shown, despite the fact that a rating model was specified in the configuration.

(7) Order of the input data:

The input data have to be provided in the following order:

- For each of the AFC models the frequencies of the *j* possible ranks of the target item (signal item) have to be provided. The first number corresponds to the number of cases with the target ranked on the first position (or selected first), the second number corresponds to the number of cases with the target ranked on the second position (or selected second), etc.
  The number of evaluated positions is given in `j.rg`. The number of alternatives, given in `k.rg`, does not determine the number of data points.
- Frequencies of the response categories for each type of stimulus for each rating SDT model are provided.

Example: Cf. Chapter 13.

3.11 Gaussian SDT model for *k*-alternative forced choice data with and without rating data including a recollection and guessing component (HTSDT.Rank) and bias parameters for modeling position bias (HTSDT.Bias.Rank)

The HTSDT-Rank model is an extension of the SDT-Rank model. It enables the modeling of *k*-alternative forced choice data from different conditions with or without rating data. Thus, the model enables one to model data from different *k*-alternative forced choice tasks. The model enables the specification of a recollection component.
The HTSDT-Bias-Rank model is an extension of the basic model that includes, in addition, parameters for modeling position bias. In the following both models are described.

3.11.1 The HTSDT-Rank model

(1) Model structure:

The probability of choosing the target alternative out of *k* alternatives in trial *j* (or alternatively the probability that target alternative has rank j $(j = 1,2,\ldots,k)$ is given by the equation:

$$\pi_j = \begin{cases} p_R + (1-p_R)\cdot\int_{-\infty}^{\infty}\left[\Phi\left(x|\mu_N,\sigma_N\right)\right]^{k-1}\cdot\phi\left(x|\mu_S,\sigma_S\right)\cdot dx & \Leftrightarrow \quad j=1 \\ (1-p_R)\cdot\int_{-\infty}^{\infty}\binom{k-1}{j-1}\cdot\left[1-\Phi\left(x|\mu_N,\sigma_N\right)\right]^{j-1}\cdot\left[\Phi\left(x|\mu_N,\sigma_N\right)\right]^{k-j}\cdot\phi\left(x|\mu_S,\sigma_S\right)\cdot dx & \Leftrightarrow \quad j=2,\ldots,k \end{cases}$$

where:

p_R	denotes the probability of recollection of an old item;
π_j	denotes the probability that the old stimulus (or target stimulus) is in position *j*;
k	represents the number of alternatives presented;
$\Phi\left(x\|\mu_N,\sigma_N\right)$	denotes the cumulative Gaussian distribution with mean μ_N and standard deviation σ_N (using standard restrictions: $\mu_N = 0$ and $\sigma_N = 1$);
$\phi\left(x\|\mu_S,\sigma_S\right)$	symbolizes the Gaussian density with mean μ_S and standard deviation σ_S.

The rating component corresponds to the DPSDT model (cf. Figure 5 on p.35)

(2) Model identification string: `"HTSDT.Rank"`

(3) Name of the file containing the model:
The source code of the model is contained in the file: `HTSDT-Rank.R`.

(4) Configuration information:

The argument `n` of the function `SDT.Estimate()` is a list containing information about the configuration of the model. This list comprises the following entries:

`n.sdt` A 2 × *n* matrix where each column represents the number of Gaussian distributions (corresponding to the number of signals) of the *DPSDT rating model* and the number of response categories.

Example:
```
n.sdt = matrix(c(2, 6, 2, 4), nr = 2)
```
indicates the presence of two rating DPSDT models, the first with 2 signals (Gaussian distributions) and 6 response categories, and the second with 2 signals and 4 response categories.

Alternatively,
```
n.sdt = c(2, 6, 2, 4)
```
can be used. The program transforms the vector to the matrix.

`k.rg` The number of alternatives from which to select the old alternative of the *ranking DPSDT model.*

`j.rg` The number of selections from the set of *k* alternatives.

`k.rg` and `j.rg` are either single numbers or, in case of more than one set of *k*-AFC data vectors of number.

If `j.rg` has not be specified, it is assumed to be equal to `k.rg`.

Example 1: `cfg = list(k.rg = 4, j.rg = 2)`

In this case only one set of forced choice data is present. The model assumes that there are two data points:

1. number of cases with the target alternative chosen as the first one;
2. number of cases with the target alterntive not chosen.

Note that `k.rg = 4` indicates that 4 alternatives are presented on each trial: the target alternative together with three distractors.

Example 2: `cfg = list(k.rg = c(4, 3, 2), j.rg = c(3, 3, 2))`

In this case three sets of forced choice data are present. The model assumes that there are eight data points (the sum of the values in `j.rg`):

1. The counts of the three possible positions of the target alternative in the first data set;
2. The counts of the three possible positions of the target alternative in the second data set;
3. The counts of the two possible positions of the target alternative in the third data set.

Note that `k.rg = c(4,3,2)` indicates that 4 alternatives are presented on each trial for the first data set (the target alternative and three distracters), 3 alternatives are presented for the second data set (the target and 2 distracters), and 2 alternatives are presented for the third data set (the target plus one distracter).

`npt = 30` The number of quadrature points for Gaussian quadrature

`restriction = "standard"` A string permitting the specification of different types of restrictions:

`"standard"`: Standard restrictions are set. These comprize the following restrictions:

1. The means of the first Gaussian distributions of each ranking and rating DPSDT model are fixed to 0.0.
2. The standard deviation parameters of the first Gaussian distributions of each ranking and rating DPSDT model are fixed to 1.0.
3. The recollection parameters associated with the first Gaussian in each rating DPSDT model is fixed to zero.

`"equal"`: In addition to the standard restrictions the mean parameters of all Gaussain models (not fixed) are set to be equal. The same is gtrue for the standard deviation parameters.

`"UVSDT"`: Restrictions are set to simulate the unequal variance SDT model.

`"EVSDT"`: Restrictions due to the equal variance SDT model are set.

`"no"`: No restrictions are set.

Comments:

- ❒ It does not matter whether capital letters or not are use. Thus `"equal"` or `"Equal"` lead to the same result.
- ❒ If no string is provided or the string does not match any of the above strings standard restrictions are set.

`rating =T` A flag indicating whether rating data should be estimated. If the flag is set to `FALSE` *k*-AFC data are modeled only (No rating data).

Example:
```
cfg = list(n.sdt = c(2, 6, 2, 4), k.rg = c(4, 3), restriction =
  "equal")
```

1. Rating data from two conditions are assumed, one with 6 and one with 4 response categories. For each condition there are 2 types of stimuli and, by consequence, to Gaussian distributions.
2. Forced choice data from 4-AFC and 3-AFC (with repeated choices or ranking) are assumed.
3. The restrictions are set to be equal.

Comment: The other entries in the list are filled in by the program resulting in:

- ❒ `j.rg = c(4, 3)`
- ❒ `npt = 30`
- ❒ `rating = T`

(5) Parameters:

The parameters of the model are passed to the function `SDT.Estimate()` in the parameter `par` in the following order (if no parameter vector is provided, the model generates a vector of starting parameters):

A. For each of the forced choice data set the model comprises the following 5 parameters:

(i) p_R^{AFC} = Probability of recollection;

(ii) μ_N^{AFC} = Mean parameter of the non-target (distracter) distribution;

(iii) σ_N^{AFC} = Variance parameter of the non-target distribution;

(iv) μ_S^{AFC} = Mean parameter of the target distribution;

(v) σ_S^{AFC} = Variance parameter of the target distribution.

B. For each Gaussian model of each DPSDT model of the rating component, the model comprises the following 3 parameters:
   (i) p^{Rating} = Probability of recollection;
   (ii) μ^{Rating} = Mean parameter of the Gaussian distribution;
   (iii) σ^{Rating} = Standard error of the Gaussian distribution;

C. For each rating SDT model $\tau_1,\ldots,\tau_r$ separating the $r+1$ response categories. These parameters follow to the recollection and Gaussian parameters of the rating DPSDT model (for each DPSDT model specified (cf. the examples).

Comment: If `rating = F`, i.e. no rating data are modeled, only parameters (i) to (v) are relevant.

Example 1: *Given:* The following configuration information:

```
cfg = list(n.sdt = c(2, 6, 2, 4), k.rg = c(4, 3), restriction = "standard")
```

The function `HTSDTParameter.Info(cfg)` provides the following output:

```
$Model
[1] "HTSDT ranking and rating model: Number of Gaussians within SDT models:
<<2,6>, <2,4>> Model types: <P,P> Number of ranking-alternatives: <4,3>
Number of ranking-positions: <4,3> Number of quadrature points: <30>
Restrictions: <standard>"
$Parameters.and.Constraints
                            name   par fixed.value ident.source Nr
1           Recollection (Rank 1)  0.00         ---          ---
2           Mean.1       (Rank 1)  0.00     0 <set>          ---
3           Stddev.1     (Rank 1)  1.00     1 <set>          ---
4           Mean.2       (Rank 1)  0.50         ---          ---
5           Stddev.2     (Rank 1)  1.00     1 <set>          ---
6           Recollection (Rank 2)  0.00         ---          ---
7           Mean.1       (Rank 2)  0.00     0 <set>          ---
8           Stddev.1     (Rank 2)  1.00     1 <set>          ---
9           Mean.2       (Rank 2)  0.50         ---          ---
10          Stddev.2     (Rank 2)  1.00     1 <set>          ---
11 [SDT-1][Gauss-1] Recollection  0.00     0 <set>          ---
12 [SDT-1][Gauss-1]         Mean  0.00     0 <set>          ---
13 [SDT-1][Gauss-1]       Stddev  1.00     1 <set>          ---
14 [SDT-1][Gauss-2] Recollection  0.00         ---          ---
15 [SDT-1][Gauss-2]         Mean  0.50         ---          ---
16 [SDT-1][Gauss-2]       Stddev  1.00     1 <set>          ---
17              [SDT-1]      c-1 -0.50         ---          ---
18              [SDT-1]      c-2 -0.25         ---          ---
19              [SDT-1]      c-3  0.00         ---          ---
20              [SDT-1]      c-4  0.25         ---          ---
21              [SDT-1]      c-5  0.50         ---          ---
22 [SDT-2][Gauss-1] Recollection  0.00     0 <set>          ---
23 [SDT-2][Gauss-1]         Mean  0.00     0 <set>          ---
24 [SDT-2][Gauss-1]       Stddev  1.00     1 <set>          ---
25 [SDT-2][Gauss-2] Recollection  0.00         ---          ---
26 [SDT-2][Gauss-2]         Mean  0.50         ---          ---
27 [SDT-2][Gauss-2]       Stddev  1.00     1 <set>          ---
28              [SDT-2]      c-1 -0.50         ---          ---
29              [SDT-2]      c-2  0.00         ---          ---
30              [SDT-2]      c-3  0.50         ---          ---
```

This reveals the order of the parameters (first column), the values of the start parameters (second column) and the fixed parameters with the fixed values (third column). There are no identity constraints.

Example 2: *Given:* The following configuration information:

```
cfg = list(n.sdt = c(2, 6, 2, 4), k.rg = c(4, 3), restriction = "equal")
```

The function `HTSDTParameter.Info(cfg)` provides the following output:

```
$Model
[1] "HTSDT ranking and rating model: Number of Gaussians within SDT models: <<2,6>,
<2,4>> Model types: <P,P> Number of ranking-alternatives: <4,3> Number of ranking-
positions: <4,3> Number of quadrature points: <30> Restrictions: <equal>"

$Parameters.and.Constraints
                           name   par fixed.value                ident.source Nr
1          Recollection (Rank 1)  0.00         ---                         ---
2          Mean.1       (Rank 1)  0.00     0 <set>                         ---
3          Stddev.1     (Rank 1)  1.00     1 <set>                         ---
4          Mean.2       (Rank 1)  0.50         ---                         ---
5          Stddev.2     (Rank 1)  1.00     1 <set>                         ---
6          Recollection (Rank 2)  0.00         --- Recollection (Rank 1) <set>  1
7          Mean.1       (Rank 2)  0.00     0 <set>                         ---
8          Stddev.1     (Rank 2)  1.00     1 <set>                         ---
9          Mean.2       (Rank 2)  0.50         --- Mean.2       (Rank 1) <set>  4
10         Stddev.2     (Rank 2)  1.00     1 <set>                         ---
11 [SDT-1][Gauss-1] Recollection  0.00     0 <set>                         ---
12 [SDT-1][Gauss-1]         Mean  0.00     0 <set>                         ---
13 [SDT-1][Gauss-1]       Stddev  1.00     1 <set>                         ---
14 [SDT-1][Gauss-2] Recollection  0.00         --- Recollection (Rank 1) <set>  1
15 [SDT-1][Gauss-2]         Mean  0.50         --- Mean.2       (Rank 1) <set>  4
16 [SDT-1][Gauss-2]       Stddev  1.00     1 <set>                         ---
17              [SDT-1]      c-1 -0.50         ---                         ---
18              [SDT-1]      c-2 -0.25         ---                         ---
19              [SDT-1]      c-3  0.00         ---                         ---
20              [SDT-1]      c-4  0.25         ---                         ---
21              [SDT-1]      c-5  0.50         ---                         ---
22 [SDT-2][Gauss-1] Recollection  0.00     0 <set>                         ---
23 [SDT-2][Gauss-1]         Mean  0.00     0 <set>                         ---
24 [SDT-2][Gauss-1]       Stddev  1.00     1 <set>                         ---
25 [SDT-2][Gauss-2] Recollection  0.00         --- Recollection (Rank 1) <set>  1
26 [SDT-2][Gauss-2]         Mean  0.50         --- Mean.2       (Rank 1) <set>  4
27 [SDT-2][Gauss-2]       Stddev  1.00     1 <set>                         ---
28              [SDT-2]      c-1 -0.50         ---                         ---
29              [SDT-2]      c-2  0.00         ---                         ---
30              [SDT-2]      c-3  0.50         ---                         ---
```

This reveals the order of the parameters (first column), the values of the start parameters (second column) and the fixed parameters with the fixed values (third column). The last columns reveals the identity constraints: The recollection and mean parameters representing the second signal of each DPSDT models (ranking and raing) are set to be equal.

Comment: Examples showing the result of setting the restrictions to `restriction = "UVSDT"` and `restriction = "EVSDT"` are found in the example file:
`HTSDT-Rank (Ex.1 Parameters & restrictions I AFC & Rating).R`

3.11.2 The HTSDT-Bias-Rank model

The HTSDT-Bias-Rank model is an extension of the HTSDT-Rank model that enables the modeling of position bias. Thus the model comprises *k* bias parameters with *k* equal to the numbers of choice alternatives presented. In addition,

(1) Model structure:

The probability of choosing the target alternative out of *k* alternatives in trial *j* (or alternatively the probability that target alternative has rank *j* $(j = 1,2,\ldots,k)$ is given by the equation:

$$\pi_j = p_G \cdot \frac{1}{k} + (1 - p_R - p_G) \cdot \int_{-\infty}^{\infty} \sum_{C \in \Gamma} \left\{ \prod_{n \in C} \left[1 - \Phi\left(x - b_n + b_j \middle| \mu_N, \sigma_N\right)\right] \cdot \prod_{m \in S \setminus C} \Phi\left(x - b_m + b_j \middle| \mu_N, \sigma_N\right) \right\} \cdot \phi\left(x \middle| \mu_S, \sigma_S\right) dx$$

$(j = 1, 2, \ldots, k)$

where:

S	is the set of *k*-1 ranks excluding rank *j*, i.e. the rank of the target alternative: $S = \{1,\ldots, j-1, j+1,\ldots,k\}$
Γ	is the set of all possible selections of *j*-1 ranks from the set *S*.
C	is a variable denoting selections in the set Γ.
b_n, b_m	are bias parameters corresponding to the positions of the ranks *n* and *m*.
b_j	is the bias parameter corresponding to the position of the target alternative.
p_R	denotes the probability of recollection of an old item;
p_G	denotes the probability of pure guessing;
π_j	denotes the probability that the old stimulus (or target stimulus) is in position *j*;
k	represents the number of alternatives presented;
$\Phi\left(x \middle| \mu_N, \sigma_N\right)$	denotes the cumulative Gaussian distribution with mean μ_N and standard deviation σ_N (using standard restrictions: $\mu_N = 0$ and $\sigma_N = 1$);
$\phi\left(x \middle| \mu_S, \sigma_S\right)$	symbolizes the Gaussian density with mean μ_S and standard deviation σ_S.

(2) Model identification string: `"HTSDT.Bias.Rank"`

(3) Name of the file containing the model:

The source code of the model is contained in the file: `HTSDT-Rank.R`.

(4) Configuration information:

Same as for HTSDT.Rank (see above, Chapter 3.11.1).

(5) Order of parameters:

The parameters of the model are passed to the function `SDT.Estimate()` in the parameter `par` in the following order (if no parameter vector is provided, the model generates a vector of starting parameters):

A. For each of the forced choice data set the model comprises the following 6 parameters:

(i) p_R^{AFC} = Probability of recollection;

(ii) p_G^{AFC} = Probability of pure guessing;

(iii) μ_N^{AFC} = Mean parameter of the non-target (distracter) distribution;

(iv) σ_N^{AFC} = Variance parameter of the non-target distribution;

(v) μ_S^{AFC} = Mean parameter of the target distribution;

(vi) σ_S^{AFC} = Variance parameter of the target distribution.

B. For each forced choice data set the model comprises, the model comprises the *k* bias parameters $b_1, b_2, \ldots, b_k$, representing position bias.

C. For each Gaussian model of the rating component, the model comprises the following 5 parameters:

(i) p_R^{SDT} = Probability of recollection;

(ii) p_G^{SDT} = Probability of pure guessing;

(iii) β^{SDT} = Guessing bias parameter indicating, in case of pure guessing, the probability selecting the most uncertain new category;

Comment: The residual probability mass of pure guessing is assigned to the most uncertain old category (in case of an even number of response categories).
In case of an uneven number of categories the the whole »guessing mass« is assigned to the middle category. In this case, the bias parameter has to be fixed to an arbitrary value between 0 and 1.

(iv) μ_N^{SDT} = Mean parameter of the non-target (distracter) distribution;

(v) σ_N^{SDT} = Variance parameter of the non-target distribution;

(vi) μ_S^{SDT} = Mean parameter of the target distribution;

(vii) σ_S^{SDT} = Variance parameter of the target distribution;

D. *r* threshold parameter $\tau_1,\ldots,\tau_r$ separating the $r+1$ response categories.

Comment: If `rating = F`, i.e. no rating data are modeled, only parameters in A and B are required.

Example: Cf. the example in the subsequent section.

(6) Types of restrictions:

Similar to the HTSDT.Rank model the HTSDT.Bias.Rank model enables the specification of three types of restrictions (standard, SDT, and extended) according to the same principles (cf. the description and examples in Chapter 3.11.1).
There is only one difference: The last bias parameter is fixed to zero. The following example illustrates the

Example (Multiple AFC models, Standard restrictions):
The following sequence of commands:

```
cfg <- list(j.rg = c(4, 3, 2), rating = F)
Erg <- SDT.Parameter.Info(n = cfg, Model.Id = "HTSDT.Bias.Rank")
print(Erg)
```

exhibits the standard fixed and equality constraint with three forced choice models:

```
$Parameters.and.Constraints
                        name par fixed.value                ident.source Nr
1  Recollection     (Rank 1)   0         ---                         ---
2  Guessing         (Rank 1)   0       0 <set>                       ---
3  Mean.1           (Rank 1)   0       0 <set>                       ---
4  Stddev.1         (Rank 1)   1       1 <set>                       ---
5  Mean.2           (Rank 1)   0         ---                         ---
6  Stddev.2         (Rank 1)   1       1 <set>                       ---
7  Bias.1           (Rank 1)   0         ---                         ---
8  Bias.2           (Rank 1)   0         ---                         ---
9  Bias.3           (Rank 1)   0         ---                         ---
10 Bias.4           (Rank 1)   0       0 <set>                       ---
```

```
11 Recollection    (Rank 2)   0         --- Recollection (Rank 1) <set>  1
12 Guessing        (Rank 2)   0     0 <set>                         ---
13 Mean.1          (Rank 2)   0     0 <set>                         ---
14 Stddev.1        (Rank 2)   1     1 <set>                         ---
15 Mean.2          (Rank 2)   0         ---       Mean.2 (Rank 1) <set>  5
16 Stddev.2        (Rank 2)   1     1 <set>                         ---
17 Bias.1          (Rank 2)   0         ---                         ---
18 Bias.2          (Rank 2)   0         ---                         ---
19 Bias.3          (Rank 2)   0     0 <set>                         ---
20 Recollection    (Rank 3)   0         --- Recollection (Rank 1) <set>  1
21 Guessing        (Rank 3)   0     0 <set>                         ---
22 Mean.1          (Rank 3)   0     0 <set>                         ---
23 Stddev.1        (Rank 3)   1     1 <set>                         ---
24 Mean.2          (Rank 3)   0         ---       Mean.2 (Rank 1) <set>  5
25 Stddev.2        (Rank 3)   1     1 <set>                         ---
26 Bias.1          (Rank 3)   0         ---                         ---
27 Bias.2          (Rank 3)   0     0 <set>                         ---
```

Comments:

- The bias parameters are denoted `Bias.1, Bias.2,` etc. and are positioned at the end of each model with the first model comprising 4, the second model 3 and the final model 2 bias parameters.
- The last bias parameter of each model is fixed to zero (e.g. Parameter #10)
- Otherwise, the same restrictions are specified in the same way as for the model without bias parameters.

(7) Order of the input data:

The input data have to be provided to the program in the following order:

- For each of the forced choice models, the frequencies of the *j* possible ranks of the target item has to be provided, for each of the *k* possible positions of the target within the set of presented alternatives:

  The first *j* numbers correspond to the *j* rankings of the target when the target item is presented in the first position.
  The second *j* numbers correspond to the *j* rankings of the target when the target item is presented in the second position, and so on.

  Thus for each model $k \cdot j$ frequencies have to be specified.

- Frequencies of the *n* response (rating) categories for the non-target class have to be provided, from the highest new confidence category to highest old confidence category;
- For each of the target classes, frequencies of the *n* response categories have to be given from the highest new confidence category to highest old confidence category.

Example (Order of input data for the HTSDT-Bias-Rank model):

Consider 2 forced choice models with 4 alternatives each. For the first model each of the possible positions of the target are specified. For the second model the counts of the last two positions of the target have been pooled together. By consequence, for the first forced choice model 16 data points have to be supplied whereas for the second forced choice model 12 data point must be supplied.
In addition, ratings with 6 response categories for old and new items are provided:

```
cfg <- list(k.rg = c(4, 4), j.rg = c(4, 3))
datavec <- c(210,  68,  68,  54,            # AFC-Model 1, Target on Position 1
             193,  98,  60,  49,            # AFC-Model 1, Target on Position 2
             198, 101,  55,  46,            # AFC-Model 1, Target on Position 3
             169,  88,  82,  61,            # AFC-Model 1, Target on Position 4

             180,  79,  141,                # AFC-Model 2, Target on Position 1
             184,  86,  130,                # AFC-Model 2, Target on Position 2
             190, 100,  110,                # AFC-Model 2, Target on Position 3
             185,  86,  129,                # AFC-Model 2, Target on Position 4

             432, 341, 309, 262, 185,  71,  # Rating data (new)
             179, 220, 238, 256, 257, 450)  # Rating data (old)
```

3.12 Gaussian mixture SDT model for modeling *k*-alternative forced choice data with and without rating data (MIX.Rank) including bias parameters for modeling position bias (MIX.Bias.Rank)

The MIX-Rank and the MIX-Bias-Rank model are another extension of the SDT-Rank model. They enable the modeling of *k*-alternative forced choice data from different conditions with or without rating data. Thus, the model enable one to model data from different *k*-alternative forced choice tasks, combined with data from a rating task using a mixture of Gaussian distributions.
Similar to the HTSDT-Rank model (cf. Chapter 3.11) there are two versions of the model: The MIX-Rank model assumes that no position bias is present, wehereas MIX-Bias-Rank comprises bias parameters for modeling position bias. In the following both models are described.

3.12.1 The MIX-Rank model

The model enables the specification of a mixture of Gaussian distributions.

(1) Model structure:

The probability of choosing the target alternative out of *k* alternatives in trial *j* (or alternatively the probability that target alternative has rank *j* $(j = 1,2,\ldots,k)$ is given by the equation:

$$\pi_j = p_{strong} \cdot \int_{-\infty}^{\infty} \binom{k-1}{j-1} \cdot \left[\Phi\left(x|\,\mu_N,\sigma_N\right)\right]^{n-j} \cdot \left[1-\Phi\left(x|\,\mu_N,\sigma_N\right)\right]^{j-1} \cdot \phi\left(x|\,\mu_{T_{strong}},\sigma_{T_{strong}}\right) \cdot dx$$

$$+ \left(1-p_{strong}\right) \cdot \int_{-\infty}^{\infty} \binom{k-1}{j-1} \cdot \left[\Phi\left(x|\,\mu_N,\sigma_N\right)\right]^{n-j} \cdot \left[1-\Phi\left(x|\,\mu_N,\sigma_N\right)\right]^{j-1} \cdot \phi\left(x|\,\mu_{T_{weak}},\sigma_{T_{weak}}\right) \cdot dx$$

where:

π_j	denotes the probability that the old stimulus (or target stimulus) is in position *j*;
k	represents the number of alternatives presented;
p_{strong}	denotes the probability of applying the strong non-target distribution;
$\Phi\left(x\|\,\mu_N,\sigma_N\right)$	denotes the cumulative Gaussian distribution with mean μ_N and standard deviation σ_N of the non-target distribution (using standard restrictions: $\mu_N = 0$ and $\sigma_N = 1$);

$\phi\left(x|\mu_{T_{strong}},\sigma_{T_{strong}}\right)$ symbolizes the Gaussian density with mean $\mu_{T_{strong}}$ and standard deviation $\sigma_{T_{strong}}$.It represents the non-target distribution resulting in high discriminability.

$\phi\left(x|\mu_{T_{weak}},\sigma_{T_{weak}}\right)$ symbolizes the Gaussian density with mean $\mu_{T_{weak}}$ and standard deviation $\sigma_{T_{weak}}$.It represents the non-target distribution resulting in low discriminability (In most applications $\mu_{T_{weak}} = 0$ and . $\sigma_{T_{weak}} = 1$).

Comment: The part of the model for modeling the rating data is identical to that of the MIX.2 model (cf. Chapter 3.4)

(2) Model identification string: `"MIX.Rank"`

(3) Name of the file containing the model:
The source code of the model is contained in the file: `MIX-Rank.R`.

(4) Configuration information:
The argument `n` of the function `SDT.Estimate()` is a list containing information about the configuration of the model. This list comprises the following entries:

`n.sdt` The number of Mixture models (corresponding to the number of different types of stimuli) making up the SDT component with each mixture model comprising two Gaussian models that make up the mixture (default: `n.sdt = 2`).

`k.rg` The number of alternatives from which to select the old alternative

`j.rg` The number of selections from the set of *k* alternatives.

`k.rg` and `j.rg` are either single numbers or, in case of more than one set of *k*-AFC data vectors of number:

Example 1: `cfg = list(k.rg = 4, j.rg = 3)`

In this case only one set of forced choice data is present. The model assumes that there are three data points:

1. number of cases with the target alternative chosen as the first one;
2. number of cases with the target alternative chosen at the second one;
3. number of cases with the target alterntive chosen as the third or fourth one.

Note that `k.rg = 4` indicates that 4 alternatives are presented on each trial: the target alternative together with three distractors.

Example 2: `cfg = list(k.rg = c(4, 3, 2), j.rg = c(3, 3, 2))`

In this case three sets of forced choice data are present. The model assumes that there are eight data points (the sum of the values in `j.rg`):

1. The counts of the three possible positions of the target alternative in the first data set;
2. The counts of the three possible positions of the target alternative in the second data set;
3. The counts of the two possible positions of the target alternative in the third data set.

Note that `k.rg = c(4,3,2)` indicates that 4 alternatives are presented on each trial for the first data set (the target alternative and three distracters), 3 alternatives are presented for the second data set (the target and 2 distracters), and 2 alternatives are presented for the third data set (the target plus one distracter).

`npt = 30` The number of quadrature points for Gaussian quadrature

`restriction = "standard"` A string permitting the specification of different types of restrictions:

- `"standard"`: Standard restrictions are set.
- `"SDT"`: Restrictions are set to simulate the unequal variance SDT model.
- `"extended"`: Restriction of an extended model allowing for unequal variances are set.
- `"no"`: No restrictions are set.

Comments:

1. Partial strings and capital letters are allowed, e.g. `"Ext"` or `"EXTENDED"` indicate extended restrictions.
2. If no string is provided or the string does not match any of the above strings standard restrictions are set.
3. Without specifying restrictions the model is not identified.

`rating =T` A flag indicating whether rating data should be estimated. If the flag is set to `FALSE` *k*-AFC data are modeled only (No rating data).

The program also accepts a single number or an incomplete list:

A single number `n` is interpreted as `k.rg = j.rg = n`. The other entries of the list assume their default values. In case of an incomplete list, the default values are always supplied by the program.

Example: `cfg = 4`

The list is internally completed by the program to:

```
cfg = list(n.sdt = 2, k.rg = 4, j.rg = 4, npt = 30, restriction =
"standard", rating = T)
```

This indicates the following setup for the `HTSDT.Rank` model:

- ❑ 2 Gaussian mixture models making up the SDT component;
- ❑ 4 alternatives are presented in the forced choice task;
- ❑ 4 rank positions (first and rest) of the target item are considered;
- ❑ 30 Gaussian quadrature points are used for evaluating the integral;
- ❑ The model with standard restrictions is set;
- ❑ *k*-AFC as well as rating data are modeled.

(5) Order of parameters:

The parameters of the model are passed to the function `SDT.Estimate()` in the parameter `par` in the following order (if no parameter vector is provided, the model generates a vector of starting parameters):

A. For each of the forced choice data set the mixture model comprises the following 7 parameters:

(i) p_{strong}^{AFC} = The probability of applying the strong target distribution (resulting in a high discriminability);

(ii) μ_{N}^{AFC} = Mean parameter of the non-target (distracter) distribution;

(iii) σ_{N}^{AFC} = Variance parameter of the non-target distribution;

(iv) $\mu_{T_{weak}}^{AFC}$ = Mean parameter of the weak target distribution;

(v) $\sigma_{T_{weak}}^{AFC}$ = Variance parameter of the weak target distribution.

(vi) $\mu_{T_{strong}}^{AFC}$ = Mean parameter of the strong target distribution;

(vii) $\sigma_{T_{strong}}^{AFC}$ = Variance parameter of the strong target distribution.

B. For each Gaussian model of the rating component, the model comprises the following 5 parameters:

(i) p_{strong}^{SDT} = The probability of applying the strong target distribution (resulting in a high discriminability);

(ii) μ_N^{SDT} = Mean parameter of the non-target (distracter) distribution;

(iii) σ_N^{SDT} = Variance parameter of the non-target distribution;

(iv) μ_T^{SDT} = Mean parameter of the target distribution.

(v) σ_T^{SDT} = Variance parameter of the target distribution

C. *r* threshold parameter $\tau_1,\ldots,\tau_r$ separating the $r+1$ response categories.

Comment: If `rating = F`, i.e. no rating data are modeled, only parameters (i) to (vii) are required.

Example: Cf. the examples in the subsequent section.

(6) Types of restrictions:

Similar to the HTSDT-Rank and HTSDT-Bias-Rank model, the MIX-Rank model enables the specification of three types of restrictions (standard, SDT, and extended) by means of providing the relevant string to the restriction option in the configuration file.
The function:

```
MIX.Rank.Restriction(cfg, fixed, ident)
```

generates a list with comprising the matrices `fixed` and `ident`. The first one corresponds to the matrix of fixed constraints whereas the second one represents the matrix of identity constraints. The function provides a convenient way to relax some of the standard constraints. One first uses the function to generate the matrices of fixed and equality constraints and then eliminates some of the constraints.
The matrices of fixed and identity constraints generated automatically are combined with the matrices of fixed and identity constraints passed in the second and third argument of the function `MIX.Rank.Restriction()`. By consequence the matrices in the output contain both the constraints generated automatically by the procedure as well as the constraints specified by the user.
However, in case of redundancy, for instance, if the same parameter is involved in a constraint specified by the user as well as in an automatically generated constraints, the specification of the user gets precedence, i.e., the parameter generated by the program is removed from the matrix of constraints whereas the user specified constraint is kept.

In the following the three different types of restrictions with different configurations are illustrated.

Configuration A: One forced choice model only:

1. Standard restrictions consist of the following fixed constraints (no equality constraints are specified):

   $\mu_N^{AFC} = 0.0$ (Mean of the non-target distribution);

   $\sigma_N^{AFC} = 1.0$ (Standard deviation of the non-target distribution);

   $\mu_{T_{weak}}^{AFC} = 0.0$ (Mean of the weak target distribution);

$\sigma_{T_{waek}}^{AFC} = 1.0$ (Standard deviation of the weak target distribution);

$\sigma_{T_{strong}}^{AFC} = 1.0$ (Standard deviation of the strong target distribution).

Example (Single AFC model, Standard restrictions):
The following sequence of commands:

```
cfg <- list(j.rg = 4, rating = F)
Erg <- SDT.Parameter.Info(n = cfg, Model.Id = "MIX.Rank")
print(Erg)
```

provides an output of the ordering and starting values of the parameters, as well as of the fixed constraints for a model containing one forced choice model and no SDT part:

```
$Parameters.and.Constraints
                name par fixed.value ident.source Nr
1 p.mix    (Rank 1) 0.0         ---          ---
2 Mean.0   (Rank 1) 0.0           0 <set>      ---
3 Stddev.0 (Rank 1) 1.0           1 <set>      ---
4 Mean.1   (Rank 1) 0.0           0 <set>      ---
5 Stddev.1 (Rank 1) 1.0           1 <set>      ---
6 Mean.2   (Rank 1) 0.5         ---          ---
7 Stddev.2 (Rank 1) 1.0           1 <set>      ---
```

Comments:

- The restrictions set as indicated by the label `<set>`.
- The values in the column `par` are the starting values of the parameters in case of no constraint being imposed on the parameter (where the value 0.0 for the mixture probability parameter `p.mix` is a raw value that is transformed to the probability $p = .5$)
- The resulting model comprises two free parameters: The recollection parameter p_{strong}^{AFC} as well as the mean $\mu_{T_{strong}}^{AFC}$ of the strong target distribution are free parameters. With 4-*AFC* data there are 3 independent data points. In this case, the model with standard restrictions can be fitted and tested.

2. SDT restrictions comprise the following set of fixed constraints:

$p_{strong}^{AFC} = 1.0$ (Probability of recollection)

$\mu_{N}^{AFC} = 0.0$ (Mean of the non-target distribution);

$\sigma_{N}^{AFC} = 1.0$ (Standard deviation of the non-target distribution);

$\mu_{T_{weak}}^{AFC} = 0.0$ (Mean of the weak target distribution);

$\sigma_{T_{waek}}^{AFC} = 1.0$ (Standard deviation of the weak target distribution);

Example (Single AFC model, SDT restrictions):
The following sequence of commands:

```
cfg <- list(j.rg = 4, rating = F, restriction = "SDT")
Erg <- SDT.Parameter.Info(n = cfg, Model.Id = "MIX.Rank")
print(Erg)
```

provides the following output:

```
$Parameters.and.Constraints
               name par fixed.value ident.source Nr
1 p.mix    (Rank 1) 0.0     1 <set>          ---
2 Mean.0   (Rank 1) 0.0     0 <set>          ---
3 Stddev.0 (Rank 1) 1.0     1 <set>          ---
4 Mean.1   (Rank 1) 0.0     0 <set>          ---
5 Stddev.1 (Rank 1) 1.0     1 <set>          ---
6 Mean.2   (Rank 1) 0.5        ---           ---
7 Stddev.2 (Rank 1) 1.0        ---           ---
```

3. Extended restrictions comprise the following set of fixed constraints:

   $\mu_N^{AFC} = 0.0$ (Mean of the non-target distribution);

   $\sigma_N^{AFC} = 1.0$ (Standard deviation of the non-target distribution);

   $\sigma_{T_{waek}}^{AFC} = 1.0$ (Standard deviation of the weak target distribution);

   $\sigma_{T_{strong}}^{AFC} = 1.0$ (Standard deviation of the weak target distribution);

   Thus, with extended restrictions the mixture probability as well as the mean of the weak and strong target distribution are free parameters.

Example (Single AFC model, Extended restrictions):

The following sequence of commands:

```
cfg <- list(j.rg = 4, rating = F, restriction = "extended")
Erg <- SDT.Parameter.Info(n = cfg, Model.Id = "MIX.Rank")
print(Erg)
```

provides the following output:

```
$Parameters.and.Constraints
               name par fixed.value ident.source Nr
1 p.mix    (Rank 1) 0.0        ---           ---
2 Mean.0   (Rank 1) 0.0     0 <set>          ---
3 Stddev.0 (Rank 1) 1.0     1 <set>          ---
4 Mean.1   (Rank 1) 0.0        ---           ---
5 Stddev.1 (Rank 1) 1.0     1 <set>          ---
6 Mean.2   (Rank 1) 0.5        ---           ---
7 Stddev.2 (Rank 1) 1.0     1 <set>          ---
```

Configuration B: Multiple forced choice models:

In case of multiple forced choice data being modeled simultaneously, equality constraints between parameters of different forced choice models are specified, additionally to the fixed constraints described above.

1. Standard restrictions comprise the following equality constraints between forced choice models:

   p_{strong}^{AFC} (Probability of recollection)

   $\mu_{T_{strong}}^{AFC}$ (Mean of the strong target distribution)

Example (Multiple AFC models, Standard restrictions):

The following sequence of commands:

```
cfg <- list(j.rg = c(4, 3, 2), rating = F)
Erg <- SDT.Parameter.Info(n = cfg, Model.Id = "MIX.Rank")
print(Erg)
```

exhibits the standard fixed and equality constraint with three forced choice models:

```
$Parameters.and.Constraints
                name par fixed.value          ident.source Nr
```

```
 1  p.mix     (Rank 1) 0.0         ---                    ---
 2  Mean.0    (Rank 1) 0.0      0 <set>                   ---
 3  Stddev.0  (Rank 1) 1.0      1 <set>                   ---
 4  Mean.1    (Rank 1) 0.0      0 <set>                   ---
 5  Stddev.1  (Rank 1) 1.0      1 <set>                   ---
 6  Mean.2    (Rank 1) 0.5         ---                    ---
 7  Stddev.2  (Rank 1) 1.0      1 <set>                   ---
-------------------------------------------------------------------------
 8  p.mix     (Rank 2) 0.0         ---  p.mix (Rank 1) <set>  1
 9  Mean.0    (Rank 2) 0.0      0 <set>                   ---
10  Stddev.0  (Rank 2) 1.0      1 <set>                   ---
11  Mean.1    (Rank 2) 0.0      0 <set>                   ---
12  Stddev.1  (Rank 2) 1.0      1 <set>                   ---
13  Mean.2    (Rank 2) 0.5         --- Mean.2 (Rank 1) <set>  6
14  Stddev.2  (Rank 2) 1.0      1 <set>                   ---
-------------------------------------------------------------------------
15  p.mix     (Rank 3) 0.0         ---  p.mix (Rank 1) <set>  1
16  Mean.0    (Rank 3) 0.0      0 <set>                   ---
17  Stddev.0  (Rank 3) 1.0      1 <set>                   ---
18  Mean.1    (Rank 3) 0.0      0 <set>                   ---
19  Stddev.1  (Rank 3) 1.0      1 <set>                   ---
20  Mean.2    (Rank 3) 0.5         --- Mean.2 (Rank 1) <set>  6
21  Stddev.2  (Rank 3) 1.0      1 <set>                   ---
```

Comments:

- The last column indicates the equality constraints. For example, Parameter 8, representing the mixture probability of the second forced choice model is set equal to Parameter 1 (representing the mixture probability of the first forced choice model. This is indicated by the entry `p.mix (Rank 1) <set>` in the next to last column in the 8th row.
  Similarly, the mixture probability parameter of the third model (Parameter 15) is set equal to the mixture probability parameter of the first forced choice model.
- Dashed lines indicate the borders between parameters from different models.
- The values under the header `par` are the values of the starting parameters.

2. SDT restrictions comprise the following equality constraints between forced choice models:

   $\mu_{T_{strong}}^{AFC}$ (Mean of the weak target distribution)

   $\sigma_{T_{strong}}^{AFC}$ (Standard deviation of the strong target distribution

Example (Multiple AFC models, SDT restrictions):
The following sequence of commands:

```
cfg <- list(j.rg = c(4, 3, 2), rating = F, restriction = "SDT")
Erg <- SDT.Parameter.Info(n = cfg, Model.Id = "MIX.Rank")
print(Erg)
```

exhibits the fixed and equality constraint for three forced choice models with SDT restrictions:

```
$Parameters.and.Constraints
                 name par fixed.value            ident.source Nr
-------------------------------------------------------------------------
1  p.mix     (Rank 1) 0.0      1 <set>                   ---
2  Mean.0    (Rank 1) 0.0      0 <set>                   ---
3  Stddev.0  (Rank 1) 1.0      1 <set>                   ---
4  Mean.1    (Rank 1) 0.0      0 <set>                   ---
5  Stddev.1  (Rank 1) 1.0      1 <set>                   ---
6  Mean.2    (Rank 1) 0.5         ---                    ---
7  Stddev.2  (Rank 1) 1.0         ---
-------------------------------------------------------------------------
```

```
 8  p.mix     (Rank 2) 0.0     1 <set>                       ---
 9  Mean.0    (Rank 2) 0.0     0 <set>                       ---
 10 Stddev.0 (Rank 2) 1.0     1 <set>                       ---
 11 Mean.1    (Rank 2) 0.0     0 <set>                       ---
 12 Stddev.1 (Rank 2) 1.0     1 <set>                       ---
 13 Mean.2    (Rank 2) 0.5        ---   Mean.2 (Rank 1) <set>  6
 14 Stddev.2 (Rank 2) 1.0        --- Stddev.2 (Rank 1) <set>  7
 15 p.mix     (Rank 3) 0.0     1 <set>                       ---
 16 Mean.0    (Rank 3) 0.0     0 <set>                       ---
 17 Stddev.0 (Rank 3) 1.0     1 <set>                       ---
 18 Mean.1    (Rank 3) 0.0     0 <set>                       ---
 19 Stddev.1 (Rank 3) 1.0     1 <set>                       ---
 20 Mean.2    (Rank 3) 0.5        ---   Mean.2 (Rank 1) <set>  6
 21 Stddev.2 (Rank 3) 1.0        --- Stddev.2 (Rank 1) <set>  7
```

3. Extended restrictions comprise the following equality constraints between forced choice models:

   p_{strong}^{AFC} (Probability of recollection)

   $\mu_{T_{weak}}^{AFC}$ (Mean of the weak target distribution)

   $\mu_{T_{strong}}^{AFC}$ (Mean of the strong target distribution)

Example (Multiple AFC models, Extended restrictions):
The following sequence of commands:

```
cfg <- list(j.rg = c(4, 3, 2), rating = F, restriction = "extended")
Erg <- SDT.Parameter.Info(n = cfg, Model.Id = "MIX.Rank")
print(Erg)
```

exhibits the standard fixed and equality constraint for three forced choice models with extended restrictions:

```
$Parameters.and.Constraints
                name par fixed.value          ident.source Nr
1  p.mix     (Rank 1) 0.0         ---                   ---
2  Mean.0    (Rank 1) 0.0     0 <set>                   ---
3  Stddev.0 (Rank 1) 1.0     1 <set>                   ---
4  Mean.1    (Rank 1) 0.0         ---                   ---
5  Stddev.1 (Rank 1) 1.0     1 <set>                   ---
6  Mean.2    (Rank 1) 0.5         ---                   ---
7  Stddev.2 (Rank 1) 1.0     1 <set>                   ---
8  p.mix     (Rank 2) 0.0         ---  p.mix (Rank 1) <set>  1
9  Mean.0    (Rank 2) 0.0     0 <set>                   ---
10 Stddev.0 (Rank 2) 1.0     1 <set>                   ---
11 Mean.1    (Rank 2) 0.0         --- Mean.1 (Rank 1) <set>  4
12 Stddev.1 (Rank 2) 1.0     1 <set>                   ---
13 Mean.2    (Rank 2) 0.5         --- Mean.2 (Rank 1) <set>  6
14 Stddev.2 (Rank 2) 1.0     1 <set>                   ---
15 p.mix     (Rank 3) 0.0         ---  p.mix (Rank 1) <set>  1
16 Mean.0    (Rank 3) 0.0     0 <set>                   ---
17 Stddev.0 (Rank 3) 1.0     1 <set>                   ---
18 Mean.1    (Rank 3) 0.0         --- Mean.1 (Rank 1) <set>  4
19 Stddev.1 (Rank 3) 1.0     1 <set>                   ---
20 Mean.2    (Rank 3) 0.5         --- Mean.2 (Rank 1) <set>  6
21 Stddev.2 (Rank 3) 1.0     1 <set>                   ---
```

Configuration C: Multiple forced choice models together with the SDT model for rating data:

In case of rating data being modeled additionally to the forced choice data the fixed and equality constraints for the forced choice models are as discussed above. In addition the following fixed constraints are added_

(a) For the first mixture model, representing the non-target stimulus class, each of the parameters is fixed, specifically:

p_{strong}^{SDT} (Probability of recollection)

μ_N^{SDT} (Mean of the non-target distribution)

σ_N^{SDT} (Standard deviation of the non-target distribution)

μ_T^{SDT} (Mean of the target distribution)

σ_T^{SDT} (Standard deviation of the target distribution)

(b) For the other Gaussian models, representing the target stimuli equality constraints are imposed according to the restrictions specified.

1. Standard restrictions comprise the following fixed as well as equality constraints between forced choice and the Gaussian (non-target) mixture models:

Fixed constraints:

$\mu_N^{SDT} = 0.0$ (Mean of the non-target distribution)

$\sigma_N^{SDT} = 1.0$ (Standard deviation of the non-target distribution)

$\sigma_T^{SDT} = 1.0$ (Standard deviation of the target distribution)

Equality constraints:

$p_{strong}^{SDT} = p_{strong}^{AFC}$ (Mixture probability)

$\mu_T^{SDT} = \mu_{T_{strong}}^{AFC}$ (Mean of the target distribution)

Example (Multiple AFC models with SDT rating model, Standard restrictions): The following sequence of commands:

```
datavec <- rep(0, 19)
cfg <- list(j.rg = c(4, 3), rating = T)
Erg <- SDT.Parameter.Info(n = cfg, Model.Id = "MIX.Rank")
print(Erg)
```

exhibits the standard fixed and equality constraint with two forced choice and two mixture SDT models:

```
$Parameters.and.Constraints
                 name   par fixed.value          ident.source Nr
1  p.mix     (Rank 1)  0.00         ---                   ---
2  Mean.0    (Rank 1)  0.00       0 <set>                 ---
3  Stddev.0  (Rank 1)  1.00       1 <set>                 ---
4  Mean.1    (Rank 1)  0.00       0 <set>                 ---
5  Stddev.1  (Rank 1)  1.00       1 <set>                 ---
6  Mean.2    (Rank 1)  0.50         ---                   ---
7  Stddev.2  (Rank 1)  1.00       1 <set>                 ---
8  p.mix     (Rank 2)  0.00         ---  p.mix (Rank 1) <set>  1
9  Mean.0    (Rank 2)  0.00       0 <set>                 ---
10 Stddev.0  (Rank 2)  1.00       1 <set>                 ---
11 Mean.1    (Rank 2)  0.00       0 <set>                 ---
12 Stddev.1  (Rank 2)  1.00       1 <set>                 ---
13 Mean.2    (Rank 2)  0.50         --- Mean.2 (Rank 1) <set>  6
14 Stddev.2  (Rank 2)  1.00       1 <set>
15 p.mix      (Mix 1)  0.00       1 <set>                 ---
16 Mean.0     (Mix 1)  0.00       0 <set>                 ---
17 Stddev.0   (Mix 1)  1.00       1 <set>                 ---
18 Mean.1     (Mix 1)  0.50       0 <set>                 ---
19 Stddev.1   (Mix 1)  1.00       1 <set>
```

```
20 p.mix     (Mix 2)  0.00         ---  p.mix (Rank 1) <set>  1
21 Mean.0    (Mix 2)  0.00     0 <set>                    ---
22 Stddev.0  (Mix 2)  1.00     1 <set>                    ---
23 Mean.1    (Mix 2)  0.50         --- Mean.2 (Rank 1) <set>  6
24 Stddev.1  (Mix 2)  1.00     1 <set>                    ---
25             c[1] -0.50         ---                    ---
26             c[2] -0.25         ---                    ---
27             c[3]  0.00         ---                    ---
28             c[4]  0.25         ---                    ---
29             c[5]  0.50         ---                    ---
```

Comments:

- In this example a pseudo data vector has to be passed to the function:
  `SDT.Parameter.Info`
  since otherwise the model is unable to compute the number of response categories for the rating model.
- Similarly to the previous example, dashed lines are used to mark separation of parameters of different models as well as of decision bounds.
- Due to the fact that the number of Gaussian SDT rating models was not specified in the configuration list (via the `n.sdt` option) the program assumes two models.

2. SDT restrictions comprise the following fixed as well as equality constraints between forced choice and the mixture SDT models:

   Fixed constraints:

   $p_{strong}^{SDT} = 1.0$ (Mixture probability)

   $\mu_N^{SDT} = 0.0$ (Mean of the non-target distribution)

   $\sigma_N^{SDT} = 1.0$ (Standard deviation of the non-target distribution)

   Equality constraints:

   $\mu_T^{SDT} = \mu_{T_{strong}}^{AFC}$ (Mean of the (strong) target distribution)

   $\sigma_T^{SDT} = \sigma_{T_{strong}}^{AFC}$ (Standard deviation of the (strong) target distribution)

   Example (Multiple AFC models with SDT rating model, SDT restrictions):
   The following sequence of commands:

```
datavec <- rep(0, 19)
cfg <- list(j.rg = c(4, 3), rating = T, restriction = "SDT")
Erg <- SDT.Parameter.Info(n = cfg, Model.Id = "MIX.Rank")
print(Erg)
```

   exhibits the fixed and equality constraint with two forced choice and two mixture SDT models with SDT restrictions:

```
$Parameters.and.Constraints
                name   par fixed.value            ident.source Nr
1  p.mix    (Rank 1)  0.00     1 <set>                    ---
2  Mean.0   (Rank 1)  0.00     0 <set>                    ---
3  Stddev.0 (Rank 1)  1.00     1 <set>                    ---
4  Mean.1   (Rank 1)  0.00     0 <set>                    ---
5  Stddev.1 (Rank 1)  1.00     1 <set>                    ---
6  Mean.2   (Rank 1)  0.50         ---                    ---
7  Stddev.2 (Rank 1)  1.00         ---                    ---
```

```
 8  p.mix     (Rank 2)  0.00     1 <set>                      ---
 9  Mean.0    (Rank 2)  0.00     0 <set>                      ---
10 Stddev.0  (Rank 2)  1.00     1 <set>                      ---
11 Mean.1    (Rank 2)  0.00     0 <set>                      ---
12 Stddev.1  (Rank 2)  1.00     1 <set>                      ---
13 Mean.2    (Rank 2)  0.50        ---   Mean.2 (Rank 1) <set>  6
14 Stddev.2  (Rank 2)  1.00        --- Stddev.2 (Rank 1) <set>  7
15 p.mix      (Mix 1)  0.00     1 <set>                      ---
16 Mean.0     (Mix 1)  0.00     0 <set>                      ---
17 Stddev.0   (Mix 1)  1.00     1 <set>                      ---
18 Mean.1     (Mix 1)  0.50     0 <set>                      ---
19 Stddev.1   (Mix 1)  1.00     1 <set>                      ---
20 p.mix      (Mix 2)  0.00     1 <set>                      ---
21 Mean.0     (Mix 2)  0.00     0 <set>                      ---
22 Stddev.0   (Mix 2)  1.00     1 <set>                      ---
23 Mean.1     (Mix 2)  0.50        ---   Mean.2 (Rank 1) <set>  6
24 Stddev.1   (Mix 2)  1.00        --- Stddev.2 (Rank 1) <set>  7
25              c[1] -0.50        ---                      ---
26              c[2] -0.25        ---                      ---
27              c[3]  0.00        ---                      ---
28              c[4]  0.25        ---                      ---
29              c[5]  0.50        ---                      ---
```

3. Extended restrictions comprise the following fixed as well as equality constraints between forced choice and the Gaussian (non-target) SDT models:

   Fixed constraints:

   $\sigma_N^{SDT} = 1.0$ (Standard deviation of the non-target distribution)

   $\sigma_T^{SDT} = 1.0$ (Standard deviation of the target distribution)

   Equality constraints:

   $p_{strong}^{SDT} = p_{strong}^{AFC}$ (Mixture probability)

   $\mu_N^{SDT} = \mu_{T_{weak}}^{AFC}$ (Mean of the (weak) target distribution)

   $\mu_T^{SDT} = \mu_{T_{strong}}^{AFC}$ (Mean of the (strong) target distribution)

Example (Multiple AFC models with SDT rating model, extended restrictions): The following sequence of commands:

```
datavec <- rep(0, 19)
cfg <- list(j.rg = c(4, 3), rating = T, restriction = "extended")
Erg <- SDT.Parameter.Info(n = cfg, Model.Id = "MIX.Rank")
print(Erg)
```

exhibits the fixed and equality constraint with two forced choice and two mixture SDT models with extended restrictions:

```
$Parameters.and.Constraints
               name   par fixed.value          ident.source Nr
1  p.mix     (Rank 1)  0.00        ---                   ---
2  Mean.0    (Rank 1)  0.00     0 <set>                  ---
3  Stddev.0  (Rank 1)  1.00     1 <set>                  ---
4  Mean.1    (Rank 1)  0.00        ---                   ---
5  Stddev.1  (Rank 1)  1.00     1 <set>                  ---
6  Mean.2    (Rank 1)  0.50        ---                   ---
7  Stddev.2  (Rank 1)  1.00     1 <set>                  ---
```

```
 8  p.mix     (Rank 2)  0.00          ---  p.mix (Rank 1) <set>  1
 9  Mean.0    (Rank 2)  0.00      0 <set>                    ---
 10 Stddev.0  (Rank 2)  1.00      1 <set>                    ---
 11 Mean.1    (Rank 2)  0.00          --- Mean.1 (Rank 1) <set>  4
 12 Stddev.1  (Rank 2)  1.00      1 <set>                    ---
 13 Mean.2    (Rank 2)  0.50          --- Mean.2 (Rank 1) <set>  6
 14 Stddev.2  (Rank 2)  1.00      1 <set>                    ---
 15 p.mix      (Mix 1)  0.00      1 <set>                    ---
 16 Mean.0     (Mix 1)  0.00      0 <set>                    ---
 17 Stddev.0   (Rank 1)  1.00      1 <set>                    ---
 18 Mean.1     (Mix 1)  0.50      0 <set>                    ---
 19 Stddev.1   (Mix 1)  1.00      1 <set>                    ---
 20 p.mix      (Mix 2)  0.00          ---  p.mix (Rank 1) <set>  1
 21 Mean.0     (Mix 2)  0.00          --- Mean.1 (Rank 1) <set>  4
 22 Stddev.0   (Mix 2)  1.00      1 <set>                    ---
 23 Mean.1     (Mix 2)  0.50          --- Mean.2 (Rank 1) <set>  6
 24 Stddev.1   (Mix 2)  1.00      1 <set>                    ---
 25               c[1] -0.50          ---                    ---
 26               c[2] -0.25          ---                    ---
 27               c[3]  0.00          ---                    ---
 28               c[4]  0.25          ---                    ---
 29               c[5]  0.50          ---                    ---
```

(7) Order of the input data:

The input data have to be provided to the program in the following order:

- For each of the forced choice models, the frequencies of the *j* possible ranks of the target item has to be provided: The first number corresponds to the number of cases with the target ranked on the first position, the second number corresponds to the number of cases with the target ranked on the second position, and so on. The number of counts has to correspond to the numbers specified in `j.rg`.
- Frequencies of the *n* response (rating) categories for the non-target class have to be provided, from the highest new confidence category to highest old confidence category;
- For each of the target classes, frequencies of the *n* response categories have to be given from the highest new confidence category to highest old confidence category.

Example: Cf. the example at the end of Chapter 3.11.1.

3.12.2 The MIX-Bias-Rank model

The MIX-Bias-Rank model is an extension of the MIX-Rank model that enables the modeling of position bias. Thus the model comprises *k* bias parameters with *k* equal to the numbers of choice alternatives presented. In addition,

(1) Model structure:

The probability of choosing the target alternative out of *k* alternatives in trial *j* (or alternatively the probability that target alternative has rank *j* $(j = 1,2,\ldots,k)$ is given by the equation:

$$\pi_j = p_{strong} \cdot \int_{-\infty}^{\infty} \sum_{C \in \Gamma} \left\{ \prod_{n \in C} \left[1 - \Phi\left(x - b_n + b_p \middle| \mu_N, \sigma_N\right) \right] \cdot \prod_{m \in S \setminus C} \Phi\left(x - b_m + b_p \middle| \mu_N, \sigma_N\right) \right\} \cdot \phi\left(x \middle| \mu_{T_{strong}}, \sigma_{T_{strong}}\right) dx$$

$$+ \left(1 - p_{strong}\right) \cdot \int_{-\infty}^{\infty} \sum_{C \in \Gamma} \left\{ \prod_{n \in C} \left[1 - \Phi\left(x - b_n + b_p \middle| \mu_N, \sigma_N\right) \right] \cdot \prod_{m \in S \setminus C} \Phi\left(x - b_m + b_p \middle| \mu_N, \sigma_N\right) \right\} \cdot \phi\left(x \middle| \mu_{T_{weak}}, \sigma_{T_{weak}}\right) dx$$

$(j = 1, 2, \ldots, k)$

where:

π_j denotes the probability that the old stimulus (or target stimulus) is in position *j*;

k represents the number of alternatives presented;

S is the set of $k-1$ positions excluding position *p*, i.e. the position of the target alternative: $S=\{1,\ldots,p-1,p+1,\ldots,k\}$

Γ is the set of all possible selections of $p-1$ positions from the set *S*.

C is a variable denoting selections in the set Γ.

p_{strong} denotes the probability of applying the strong non-target distribution;

$\Phi(x|\mu_N,\sigma_N)$ denotes the cumulative Gaussian distribution with mean μ_N and standard deviation σ_N of the non-target distribution (using standard restrictions: $\mu_N=0$ and $\sigma_N=1$);

$\phi(x|\mu_{T_{strong}},\sigma_{T_{strong}})$ symbolizes the Gaussian density with mean $\mu_{T_{strong}}$ and standard deviation $\sigma_{T_{strong}}$.It represents the non-target distribution resulting in high discriminability.

$\phi(x|\mu_{T_{weak}},\sigma_{T_{weak}})$ symbolizes the Gaussian density with mean $\mu_{T_{weak}}$ and standard deviation $\sigma_{T_{weak}}$.It represents the non-target distribution resulting in low discriminability (In most applications $\mu_{T_{weak}}=0$ and . $\sigma_{T_{weak}}=1$).

b_n , b_m are bias parameters corresponding to the positions of the ranks *n* and *m*.

b_p is the bias parameter corresponding to the position of the target alternative.

(2) Model identification string: `"MIX.Bias.Rank"`

(3) Name of the file containing the model:
The source code of the model is contained in the file: `MIX-Bias-Rank.R`.

(4) Configuration information:
Same as for MIX.Rank (see Chapter 3.12.1).

(5) Order of parameters:
The parameters of the model are passed to the function `SDT.Estimate()` in the parameter `par` in the following order (if no parameter vector is provided, the model generates a vector of starting parameters):

A. For each of the forced choice data set the model comprises the following 7 parameters:

(i) p_{strong}^{AFC} = The probability of applying the strong target distribution (resulting in a high discriminability);

(ii) μ_N^{AFC} = Mean parameter of the non-target (distracter) distribution;

(iii) σ_N^{AFC} = Variance parameter of the non-target distribution;

(iv) $\mu_{T_{weak}}^{AFC}$ = Mean parameter of the weak target distribution;

(v) $\sigma_{T_{weak}}^{AFC}$ = Variance parameter of the weak target distribution;

(vi) $\mu_{T_{strong}}^{AFC}$ = Mean parameter of the strong target distribution;

(vii) $\sigma_{T_{strong}}^{AFC}$ = Variance parameter of the strong target distribution.

B. For each forced choice data set the model comprises, the model comprises the *k* bias parameters $b_1, b_2, \ldots, b_k$, representing position bias.

C. For each Gaussian mixture model of the rating component, the model comprises the following 5 parameters:

(i) p_{strong}^{SDT} = The probability of applying the strong target distribution (resulting in a high discriminability);

(ii) μ_N^{SDT} = Mean parameter of the non-target (distracter) distribution;

(iii) σ_N^{SDT} = Variance parameter of the non-target distribution;

(iv) μ_T^{SDT} = Mean parameter of the target distribution;

(v) σ_T^{SDT} = Variance parameter of the target distribution.

(vi) *r* threshold parameter $\tau_1, \ldots, \tau_r$ separating the $r+1$ response categories.

Comment: If `rating = F`, i.e. no rating data are modeled, only parameters (i) to (vii) are required.

Example: Cf. the examples in the subsequent section.

(6) Types of restrictions:

Similar to the MIX.Rank model the MIX.Bias.Rank model enables the specification of three types of restrictions (standard, SDT, and extended) with the same principles (cf. the description and examples in Chapter 3.12.1).

There is only one difference: The last bias parameter is fixed to zero. The following example illustrates the

Example (Multiple AFC models, Standard restrictions):

The following sequence of commands:

```
cfg <- list(j.rg = c(4, 3, 2), rating = F)
Erg <- SDT.Parameter.Info(n = cfg, Model.Id = "MIX.Bias.Rank")
print(Erg)
```

exhibits the standard fixed and equality constraint with three forced choice models:

```
$Parameters.and.Constraints
                name par fixed.value          ident.source Nr
1  p.mix    (Rank 1) 0.0         ---                   ---
2  Mean.0   (Rank 1) 0.0     0 <set>                   ---
3  Stddev.0 (Rank 1) 1.0     1 <set>                   ---
4  Mean.1   (Rank 1) 0.0     0 <set>                   ---
5  Stddev.1 (Rank 1) 1.0     1 <set>                   ---
6  Mean.2   (Rank 1) 0.5         ---                   ---
7  Stddev.2 (Rank 1) 1.0     1 <set>                   ---
8  Bias.1   (Rank 1) 0.0         ---                   ---
9  Bias.2   (Rank 1) 0.0         ---                   ---
10 Bias.3   (Rank 1) 0.0         ---                   ---
11 Bias.4   (Rank 1) 0.0     0 <set>                   ---
12 p.mix    (Rank 2) 0.0         ---  p.mix (Rank 1) <set>  1
13 Mean.0   (Rank 2) 0.0     0 <set>                   ---
14 Stddev.0 (Rank 2) 1.0     1 <set>                   ---
15 Mean.1   (Rank 2) 0.0     0 <set>                   ---
16 Stddev.1 (Rank 2) 1.0     1 <set>                   ---
17 Mean.2   (Rank 2) 0.5         --- Mean.2 (Rank 1) <set>  6
18 Stddev.2 (Rank 2) 1.0     1 <set>                   ---
19 Bias.1   (Rank 2) 0.0         ---                   ---
20 Bias.2   (Rank 2) 0.0         ---                   ---
21 Bias.3   (Rank 2) 0.0     0 <set>                   ---
```

```
22 p.mix    (Rank 3) 0.0         ---  p.mix (Rank 1) <set>  1
23 Mean.0   (Rank 3) 0.0     0 <set>                   ---
24 Stddev.0 (Rank 3) 1.0     1 <set>                   ---
25 Mean.1   (Rank 3) 0.0     0 <set>                   ---
26 Stddev.1 (Rank 3) 1.0     1 <set>                   ---
27 Mean.2   (Rank 3) 0.5         --- Mean.2 (Rank 1) <set>  6
28 Stddev.2 (Rank 3) 1.0     1 <set>                   ---
29 Bias.1   (Rank 3) 0.0         ---                   ---
30 Bias.2   (Rank 3) 0.0     0 <set>                   ---
```

Comments:

- The bias parameters are denoted `Bias.1, Bias.2,` etc. and are positioned at the end of each model with the first model comprising 4, the second model 3 and the final model 2 bias parameters.
- The last bias parameter of each model is fixed to zero (e.g. Parameter #11)
- Otherwise, the same restrictions are specified in the same way as for the model without bias parameters.

(7) Order of the input data:

The input data have to be provided to the program in the following order:

- For each of the forced choice models, the frequencies of the *j* possible ranks of the target item has to be provided, for each of the *k* possible positions of the target within the set of presented alternatives:

  The first *j* numbers correspond to the *j* rankings of the target when the target item is presented in the first position.
  The second *j* numbers correspond to the *j* rankings of the target when the target item is presented in the second position, and so on.

  Thus for each model $k \cdot j$ frequencies have to be specified.

- Frequencies of the *n* response (rating) categories for the non-target class have to be provided, from the highest new confidence category to highest old confidence category;
- For each of the target classes, frequencies of the *n* response categories have to be given from the highest new confidence category to highest old confidence category.

Example (Order of input data for the MIX-Bias-Rank model):

Consider 2 forced choice models with 4 alternatives each. For the first model each of the possible positions of the target are specified. For the second model the counts of the last two positions of the target have been pooled together. By consequence, for the first forced choice model 16 data points have to be supplied whereas for the second forced choice model 12 data point must be supplied.
In addition, ratings with 6 response categories for old and new items are provided:

```
cfg <- list(k.rg = c(4, 4), j.rg = c(4, 3))
datavec <- c(210,  68,  68,  54,            # AFC-Model 1, Target on Position 1
             193,  98,  60,  49,            # AFC-Model 1, Target on Position 2
             198, 101,  55,  46,            # AFC-Model 1, Target on Position 3
             169,  88,  82,  61,            # AFC-Model 1, Target on Position 4

             180,  79,  141,                # AFC-Model 2, Target on Position 1
             184,  86,  130,                # AFC-Model 2, Target on Position 2
             190, 100,  110,                # AFC-Model 2, Target on Position 3
             185,  86,  129,                # AFC-Model 2, Target on Position 4

             432, 341, 309, 262, 185,  71,  # Rating data (new)
             179, 220, 238, 256, 257, 450)  # Rating data (old)
```

3.13 Gaussian SDT model with probabilistic response functions: GRM-SDT and ORM-SDT

The GRM-SDT (SDT model with graded response function) and the ORM-SDT (SDT with ordinal Rasch response function) employ probabilistic response models instead of the deterministic one of the common SDT model. The latent distributions of the decision variables are the same as for the Gaussian SDT model, i.e. normal distributions.

(1) Model structure:

Two different response models are implemented:

(a) Samejima's (1969, 1997, 2010) graded response model:

The response process is characterized by the following equations:

The conditional probability of selecting a response category greater or equal to k $(k = 1,\ldots,K)$, given a fixed value of the latent decision variable is given by the following equation:

$$P^*\left(R \geq k \middle| \theta\right) = \frac{\exp\left[\alpha \cdot \left(\theta - \tau_{k-1}\right)\right]}{1 + \exp\left[\alpha \cdot \left(\theta - \tau_{k-1}\right)\right]},$$

Where R represents the given response, θ denotes the value on the latent decision axis (latent signal strength), α is a slope parameter (discrimination parameter), and τ_{k-1} is a difficulty parameter for selecting a response category greater or equal to k. Similar to the thresholds separating response regions in SDT models, the $K-1$ difficulty parameters are ordered: $\tau_1 \leq \tau_2 \leq \cdots \leq \tau_{K-1}$.

The grader response model replaces the step functions of the traditional SDT model:

$$P^*\left(R \geq k \middle| \theta\right) = \begin{cases} 1 & \Leftrightarrow \quad \theta \geq \tau_{k-1} \\ 0 & \Leftrightarrow \quad \theta < \tau_{k-1} \end{cases}$$

by the smoother sigmoidal function of the above equation. With $\alpha \to \infty$ the graded response model of the first equation turns into the deterministic response process, given by the second equation of the standard SDT model for ordinal responses (cf. Macmillan & Creelman, 2005; Wickens, 2002).

The conditional probability of selecting response category *k* results from the difference between the conditional probabilities given by Equation 1:

$$P\left(R = k \middle| \theta\right) = \begin{cases} 1 - P^*\left(R \geq k+1 \middle| \theta\right) & \Leftrightarrow \quad k = 1 \\ P^*\left(R \geq k \middle| \theta\right) - P^*\left(R \geq k+1 \middle| \theta\right) & \Leftrightarrow \quad k = 2,3,\ldots,K-1 \\ P^*\left(R \geq k \middle| \theta\right) & \Leftrightarrow \quad k = K \end{cases}.$$

(b) The ordinal Rasch model based on the partial credit model (Masters, 1982, 2010; Masters & Wright, 1984; Muraki, 1992):

This conditional probability is provided by the following equation:

$$P^*\left(R = k+1 \middle| \theta, R \in \{k, k+1\}\right) = \frac{\exp\left[\alpha_k \cdot \left(\theta - \tau_k\right)\right]}{1 + \exp\left[\alpha_k \cdot \left(\theta - \tau_k\right)\right]},$$

where *R* represents the given response, θ denotes the value on the latent decision axis, α_k is a slope parameter (discrimination parameter) associated with the transition from response category k to $k+1$. Finally, τ_k is a difficulty parameter related to the transition from response category k to $k+1$. The probability of selecting response category *k* is given be the following equation (for a derivation, see, for example, Masters, 1982):

$$P\left(R=k \mid \theta\right)=\begin{cases} \dfrac{1}{1+\sum_{l=2}^{K}\left[\exp\left(\sum_{i=2}^{l}\alpha_{i-1}\cdot\left(\theta-\tau_{i-1}\right)\right)\right]} & \Leftrightarrow \quad k=1 \\ \dfrac{\exp\left(\sum_{i=2}^{k}\alpha_{i-1}\cdot\left(\theta-\tau_{i-1}\right)\right)}{1+\sum_{l=2}^{K}\left[\exp\left(\sum_{i=2}^{l}\alpha_{i-1}\cdot\left(\theta-\tau_{i-1}\right)\right)\right]} & \Leftrightarrow \quad k=2,3,\ldots,K \end{cases}.$$

Contrary to the graded response model, the step difficulty parameters $\tau_1,\tau_2,\ldots,\tau_{K-1}$ need not be ordered. In addition, the discrimination parameters $\alpha_1,\alpha_2,\ldots,\alpha_{K-1}$ need not be the same.

The unconditional probability of $P\left(R=k\right)$ is given by taking the expectation with respect to the distribution of the latent decision variable θ:

$$\pi_k=P\left(R=k\right)=\mathrm{E}\left[P\left(R=k \mid \theta\right)\right]=\int_{-\infty}^{\infty}P\left(R=k \mid \theta\right)\cdot\phi\left(\theta \mid \mu,\sigma^2\right)\,\mathrm{d}\theta,$$

where $\phi\left(\theta \mid \mu,\sigma^2\right)$ denotes the density function of the normal distribution with mean μ and variance σ^2.

The integration is performed numerically using Gauss-Hermite quadrature.

(2) Model identification string: `"IRF.Gauss"`

This string can be passed to the function `SDT.Estimate()` with the argument `Model.Id` (cf. Chapter 1).

(3) Name of the file containing the model:

The source code of the model is contained in the file: `SDT-IRF-Gauss.R`.

(4) Configuration information:

The configuration information is passed to the function `SDT.Estimate()` in the argument `n`. The configuration information consists of a list with the following entries:

`n.sdt` = The number of signals (default: `n.sdt = 2`)

`model` = Type of probabilistic item response model:
ORM = Ordinal Rasch model (partial credit model)
GRM = Graded response model
[Default = "ORM"]

`restriction` = A string specifying the type of restrictions: The following options are available:

`"NO"`	No restrictions
`"STANDARD"`	♦ The mean and variance parameter of the first Gaussian are fixed to $\mu=0$ and $\sigma=1$. ♦ The discrimination parameters are all fixed to 1. (Default option)
`"STANDARD-EQUAL-A"`	♦ The mean and variance parameter of the first Gaussian are fixed to $\mu=0$ and $\sigma=1$. ♦ The discrimination parameters are set to be equal.

`"EQUAL-A"`	The discrimination parameters are set to be equal.
`"EQUAL-S"`	♦ The mean parameter of the first Gaussian model is equal to 0. ♦ The standard deviation parameters are set to be equal to 1.0 for all Gaussian models.
`"EQUAL-AS"`	♦ The mean parameter of the first Gaussian model is equal to 0. ♦ The standard deviation parameters are set to be equal to 1.0 for all Gaussian models. ♦ The discrimination parameters are set to be equal.
`"EQUAL-S-FIX-A"`	♦ The mean parameter of the first Gaussian model is equal to 0. ♦ The standard deviation parameters are set to be equal to 1.0 for all Gaussian models. ♦ The discrimination parameters are fixed at 1.0.

`npt` = Number of quadrature points for Gauss-Hermite quadrature (default = 35).
Comment: In case of great values of the discrimination parameters it is useful to use much more quadrature points.

`robust` = TRUE or FALSE. In case of TRUE the ordering of the thresholds is guaranteed (only relevant with model GRM). [default = FALSE]

Example:

```
cfg <- list(n.sdt = 4, restriction = "STANDARD-EQUAL-A")
```

tells the estimation function that there are 4 types of signals and the specified restrictions are the standard restrictions on the Gaussian models as well as equal discrimination parameters for each threshold (see above).
The resulting configuration list `cfg` looks like this:

```
$n.sdt
[1] 4
$restriction
[1] "STANDARD-EQUAL-A"
$npt
[1] 35
$model
[1] "ORM"
$robust
[1] FALSE
```

(5) Order of parameters:

Parameters of the model are in the following order (passed to the function `SDT.Estimate()` in the parameter `par`):

I. Two parameters characterizing the Gaussian signal distributions repeated for each signal distribution $j = 1, 2, 3, ..., n$.

(iii) μ_j = Mean of the Gaussian model representing signal distribution *j*.

(iv) σ_j = Standard deviation of the Gaussian model representing signal distribution *j*.

II. $t_1, t_2, ..., t_{R-1}$, = thresholds (decision bounds), where *R* denotes the number of response categories.

III. $a_1, a_2, ..., a_{R-1}$, = discrimination parameters, where *R* denotes the number of response categories.

Help/Tip:

The function:

```
SDT.Parameter.Info(data = NULL, par = NULL, n = 2, Model.Id =
"SDT", fixed = NULL, ident = NULL, deci = 3)
```

displays the parameter configuration (a description of the parameters of the function is given in Chapter 2.4):

Example:

Given: The data of Ratcliff et al. (1994), Experiment 1, pure strong items (2 types of signals with 6 response categories per signal):

```
datavec <- c(477, 776, 527, 321, 258, 184,      # Pure Strong New
             192, 401, 290, 267, 316, 442)      # Pure Strong Old
```

The sequence of commands:

```
cfg <- list(n.sdt = 2)
PI <- SDT.Parameter.Info(data = datavec, n = cfg, Model.Id =
"IRF.Gauss")
print(PI)
```

results in the following output:

```
$Model
[1] "Gaussian SDT model with probabilistic item response functions
<Model = ORM>, <Restriction = STANDARD>, <Robust thresholds =
FALSE>, <Number of quadrature points = 35>"

$Parameters.and.Constraints
             name   par fixed.value ident.source Nr
1  [SDT-1]   Mean  0.00     0 <set>          ---
2  [SDT-1] Stddev  1.00     1 <set>          ---
3  [SDT-2]   Mean  0.00         ---          ---
4  [SDT-2] Stddev  1.00         ---          ---
5             c[1] -0.50         ---          ---
6             c[2] -0.25         ---          ---
7             c[3]  0.00         ---          ---
8             c[4]  0.25         ---          ---
9             c[5]  0.50         ---          ---
10            a[1]  1.00     1 <set>          ---
11            a[2]  1.00     1 <set>          ---
12            a[3]  1.00     1 <set>          ---
13            a[4]  1.00     1 <set>          ---
14            a[5]  1.00     1 <set>          ---
```

Comment: Standard restrictions are set:

- The mean and the standard deviation of the first signal distribution are fixed to 0 and 1.
- The discrimination parameters are fixed to 1.0.

(6) Order of input data:
The input data are response frequencies for the different response categories. The data for the noise signal are presented first, followed by the data for the other signals. The order of the data within each signal is from the highest noise (new) category (e.g. "sure noise" or "sure new") to the highest signal (old) category (e.g. "sure signal" or "sure old").

Comment: The output presents the data in the same order.

(7) Model functions:
The functions for computing model probabilities of the graded response model and the ordinal Rasch model are, respectively:

```
IRF.GRM.Gauss(parvec, cfg, fixed = NULL, ident = NULL)
IRF.ORM.Gauss(parvec, cfg, fixed = NULL, ident = NULL)
```

`parvec`	vector of parameters
`cfg`	list with configuration information (see above).
`fixed`	matrix of fixed constraints (not used).
`ident`	matrix of identity constraints (not used).

Comment: The function can be used in isolation, e.g. for generating artificial data.

(8) Model matrix (analytical computation):
The model matrix (i.e., a $m \times p$ matrix of partial derivatives of the model probabilities with respect to the free parameters [m = number of generated probabilities (= number of data points), p = number of free model parameters] is computed by the following function:

```
IRF.Gauss.Model.Matrix(full.par, cfg, fixed = NULL, ident = NULL,
functional = NULL)
```

`full.par`	Full vector of parameters
`cfg`	List with configuration information (see above).
`fixed`	$2 \times q$ matrix of fixed constraints (cf. Chapter 4.1)
`ident`	$2 \times r$ matrix of equality constraints (cf. Chapter 4.2)
`functional`	Function implementing functional constraints. The function has to contain the attribute `"SDT.Jacobian"` containing a function for computing the Jacobian matrix of the functional constraints. If this function is not available in case of functional constraints being specified, the function computing the model matrix returns the value `NULL`. (For further details, see Chapter 4.4)

Comment: The function can be used in isolation, e.g. for specific tests of the model.

(9) Model matrix (numerical computation):
The following function computes the model matrix numerically, using the function `jacobian()` from the package `NumDeriv`.

```
IRF.Gauss.Matrix.Num(full.par, cfg, fixed = NULL, ident = NULL,
functional = NULL)
```

`full.par`	Full vector of parameters
`cfg`	List with configuration information (see above).
`fixed`	$2 \times q$ matrix of fixed constraints (cf. Chapter 4.1)
`ident`	$2 \times r$ matrix of equality constraints (cf. Chapter 4.2)
`functional`	Function implementing functional constraints.

Comments:

- Contrary to the analytical counterpart this function does not require the specification of a function computing the Jacobian matrix in case of functional constraints being specified.
- The function is invoked for computing the model matrix in the function `SDT.Statistics()` in the following situations:
  (i) Functional constraints have been specified by the user without specifying a function for computing the Jacobian of the constraints.
  (ii) The flag `sym.gr` was set to `FALSE` in the estimation procedure thus preventing the computation of symbolic derivatives.

4. Imposing constraints on parameters

The following types of constraints on parameters may be specified:

1. Parameters may be fixed to specific values.
2. Equality constraints may be imposed on them.
3. Arbitrary functional constraints may be imposed on parameters.

Fixed and equality constraints may be specified by means of matrixes that are passed to the estimation function `SDT.Estimate()` (cf., Chapter 2.1, on Page 2). Functional constraints (as well as fixed and equality constraints) are specified by means of a user-defined function that is passed to the estimation function.
In the following, the three methods are considered in detail.

4.1 Fixing values of parameters

Values of parameters are fixed by specifying a matrix with two rows:

Row 1: Contains the *values* of the parameters to be fixed,
Row 2: Contains the *positions* of the parameter to be fixed.

Example:

Given: The vector of starting parameters for the SDT model with 4 types of signals and 6 response categories per signal (cf. Chapter 3.1 for a description of the order and meaning of parameters of the SDT model):

```
#          m2  std2    m3  std3    m4  std4     t1    t2    t3    t4    t5
par <- c(0.00, 1.00, 0.00, 1.00, 0.00, 1.00, -1.50,-0.75, 0.00, 0.75, 1.50)
```

Assume that we want to fix the scale parameters `std2`, `std3`, and `std4` (i.e. the standard deviations of the Gaussians associated with signals 2, 3 and 4, respectively) to 1. We proceed as follows:

First, the positions of the fixed parameters within the parameter vector are specified:

```
pos <- c(2, 4, 6)
```

Second the 2×3 matrix with the fixed values of parameters the in the first and positions of the parameters in the second row is generated:

```
fixed <- matrix(c(par[pos], pos), nrow = 2, byrow = T)
```

For the present example, the resulting matrix looks like this:

$$\text{fixed} = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 4 & 6 \end{bmatrix}$$

This matrix is passed to the estimation procedure `SDT.Estimate()` in the argument `fixed`.

4.1.1 Possible errors in specifying fixed constraints

The estimation routine checks for multiple occurrence of a parameter in the second row of the matrix of fixed constraints. In this case an error message is emitted and the program stops.

Example:

The following specification of fixed constraints:

```
fixed <- matrix(c(0, 1, 0, 1, 1, 2, 1, 2), nr = 2, byrow = T)
```

resulting in the matrix:

$$\text{fixed} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 2 & 1 & 2 \end{bmatrix},$$

leads to the following error message:

```
FATAL ERROR: Duplicated fixed constraints found on the following positions:  3 4
Error in SDT.Estimate(data = data, fixed = fixed, ident = ident, test = T):
```

Note that the error message indicates the positions with duplicated parameters (the positions 3 and 4, in the present case).

4.2 Imposing equality constraints on parameters

Equality constraints on parameters are imposed by specifying a matrix with two rows:

Row 1: Contains the positions of the first parameter (*source parameters*);
Row 2: Contains the positions of the second parameters (*target parameters*) that are equated to the ones in the first row on the same position.

Example:
Given: The vector of starting parameters for the SDT model with 4 types of signals and 6 response categories per signal (cf. Chapter 3.1 for a description of the order and meaning of parameters):

```
#          m2  std2    m3  std3    m4  std4     t1    t2    t3    t4    t5
par <- c(0.00, 1.00, 0.00, 1.00, 0.00, 1.00, -1.50,-0.75, 0.00, 0.75, 1.50)
```

Assume that we want to restrict the scale parameters `std2`, `sts3` and `sts4` (i.e. the standard deviations of the Gaussians associated with signals 2, 3 and 4, respectively) to be equal. The matrix of equality constraints may be specified as follows:

```
pos.source <- c(2, 2)
pos.target <- c(4, 6)
ident <- matrix(c(pos.source, pos.target), nrow = 2, byrow = T)
```

The resulting matrix looks like this:

$$\text{ident} = \begin{bmatrix} 2 & 2 \\ 4 & 6 \end{bmatrix}$$

This matrix tells the program that the parameter on the forth position (`std3`) has to be equated to that on the second position (`std2`), and, similarly, the parameter on the sixth position (`std4`) has to be equated to that on the second position (`std2`).

4.2.1 Possible errors in specifying equality constraints

One possible error in the specification of equality constraints consists in putting the same index (of a parameter) in both rows of the matrix of equality constraints.

Example:
The following specification:

```
ident <- matrix(c(1, 2, 3, 5, 1, 3, 5, 6), nr = 2, byrow = T)
```

resulting in the matrix:

$$\text{ident} = \begin{bmatrix} 1 & 2 & 3 & 5 \\ 1 & 3 & 5 & 6 \end{bmatrix}$$

leads to the following error message:

```
FATAL ERROR: Parameters subjected to identity constraints on the following source
positions:  1 3 4
are also found on target positions
Error in SDT.Estimate(data = data, fixed = fixed, ident = ident, test = T) :
```

The error message indicates the positions of those source parameters that function also as target parameters (in the present case, the parameters are in the positions 1, 3 and 4).

4.2.2 Restrictions on equality constraints for specific models

4.2.2.1 MODEL HT.N

- The model *HT.n* checks for the existence of equality constraints between rating probability parameters and other types of parameters. If this is the case an error message occurs and the program stops.

Example 1: A mean and a recollection probability parameter were used as source with rating probability parameters as targets of the equality constraints. This resulted in the following error message:

```
Error in SDT.HT.n.check.identity.constraints(par, n, ident):
FATAL ERROR (HT.n): Equality constraint between rating parameters and other types
of parameters:
 Source Parameter: mean[2] p[2]
```

Example 2: A mean and a standard deviation parameter were used as target with rating probability parameters as sources of the equality constraints. This resulted in the following error message:

```
Error in SDT.HT.n.check.identity.constraints(par, n, ident):
FATAL ERROR (HT.n): Equality constraint between rating parameters and other types
of parameters:
 Target Parameter: mean[2] stddev[2]
```

- The model *HT.n* also checks for the existence of equality constraints between recollection probability parameters and other types of parameters. If this is the case, an error message occurs too, and the program stops.

Example 3: A mean parameter was used as the source with a recollection probability parameter as the target of the equality constraint. This resulted in the following error message:

```
Error in SDT.HT.n.check.identity.constraints(par, n, ident):
```

```
FATAL ERROR (HT.n): Equality constraint between recollection parameters and other
types of parameters:
 Source Parameter: mean[2]
```

Example 4: A mean parameter was the target with a recollection probability parameter as the source of the equality constraints. This resulted in the following error message:

```
Error in SDT.HT.n.check.identity.constraints(par, n, ident):
FATAL ERROR (HT.n): Equality constraint between recollection parameters and other
types of parameters:
 Target Parameter: mean[2]
```

- The model does not allow for the specification of equality constraints between rating parameters of different signals, if the source parameter belongs to a signal whose parameters are subjected to between-signals equality constraints (This problem can only occur for models comprising more than two signals).

Example 5: The following structure, generated by the function `SDT.Parameter.Info()` [cf. Chapter 2.4], gives an overview of the fixed and equality constraints of the model in question:

```
        name    par fixed.value ident.source Nr
1       p[1]  0.500         ---          ---
2     p-1[1]  0.000           0          ---
3     p-2[1]  0.089         ---          ---
4     p-3[1]  0.243         ---          ---
5     p-4[1]  0.488         ---          ---
6     p-5[1]  0.180 Redundant-p          ---
7    mean[2]  0.000         ---          ---
8  stddev[2]  1.000         ---          ---
9       p[2]  0.500         ---          ---
10    p-1[2]  0.400         0.4          ---
11    p-2[2]  0.090        0.09          ---
12    p-3[2]  0.089         ---       p-2[1]  3
13    p-4[2]  0.210         ---       p-3[2] 12
14    p-5[2]  0.210 Redundant-p          ---
15   mean[3]  0.000         ---          ---
16 stddev[3]  1.000         ---          ---
17      p[3]  0.500         ---          ---
18    p-1[3]  0.233 Redundant-p          ---
19    p-2[3]  0.089         ---       p-2[1]  3
20    p-3[3]  0.210         ---       p-5[2] 14
21    p-4[3]  0.233         ---       p-1[3] 18
22    p-5[3]  0.233         ---       p-1[3] 18
23       t-1 -1.500         ---          ---
24       t-2 -0.500         ---          ---
25       t-3  0.500         ---          ---
26       t-4  1.500         ---          ---
```

The column labeled `ident.source` contains the names of the source parameters of the identity constraints. For example, the third rating probability parameter of Model 2 (`p-3[2]`) is equated to the second rating probability parameter of Model 1 (`p-2[1]`).

This configuration results in the following error message:

```
FATAL ERROR: Not allowed equality constraints between signals
specified:
   Number of Signal (target):  3
   Number of Signal (source):  2
The source of the constraint is located within a model some of whose
parameters are itself subjected to equality constraints.
In order to fit the model with this type of constraint, set: sym.gr = F
in the estimation procedure SDT.Estimate().
```

that is followed by a halt of program.

This error message is caused by the constraints `p-3[2] = p-2[1]` (Line 12) and `p-3[3] = 14` (Line 20). Due to these two constraints a between-signals constraint between rating probability parameters of Signal 2 and 3 is specified, with a probability parameter of Signal 2 being, at the same time, subjected to a between-signals constraint.

The reason for not allowing these types of constraints lies in the fact that the computation of the Jacobian for the transformation of raw to probability parameters may become extremely complex if these types of constraints are permitted.

As indicated in the error message, models with these types of constraints can be fitted by setting the flag `sym.gr = F` in the estimation procedure `SDT.Estimate()`. In this case, no symbolic derivates are computed resulting in an increase of the time needed for performing the estimation (cf. Chapter 2.1).

4.3 Resolution of conflicting constraints

If a parameter is subjected to a fixed as well as to an equality constraint, then all parameters specified as being equal to the fixed parameter are transformed to fixed parameters by the estimation procedure.
In addition, the routine performing the resolution of conflicting constraints eliminates duplicate parameters from the matrix of fixed constraints.

4.4 Specifying complex (nonlinear) functional constraints on parameters

The models enable the specification of functional constraints on parameters. In order to specify these complex functional constraints on parameters the user has to define a function that receives as input a vector of parameters and returns a new vector of parameters containing the functional constraints.
In addition, a gradient function performing the adjustment of the gradient vector has to be specified that takes the functional constraints into account. Formally, the adjustment of the gradient amounts to a multiplication of the gradient vector $\nabla_{\boldsymbol{\theta}}$ with a Jacobian matrix **J** that contains the partial derivatives of the functional constraints of the (old) parameters with respect to the new parameters.

4.4.1 Basic principles

In the following, a formal exposition of the procedure for computing the new gradient vector is provided. Then the implementation of functional constraints and the adjustment of the gradient within the models is provided.

Given:

$\boldsymbol{\theta} = [\theta_1, \theta_2, \cdots \theta_n]^{\mathbf{T}}$ — The original vector of parameters without functional constraints;

$\nabla_{\boldsymbol{\theta}} = \left[\frac{\partial \log L}{\partial \theta_1}, \frac{\partial \log L}{\partial \theta_2}, \cdots, \frac{\partial \log L}{\partial \theta_n} \right]^{\mathbf{T}}$ — The gradient vector, i.e., the vector of partial derivatives of the log likelihood function with respect to the original parameters.

For ease of exposition we assume that the parameters within the parameter and gradient vector are ordered in such a way that the *m source parameters* appear before (*n*-*m*) *target parameters*.

We now consider the parameter vector $\boldsymbol{\theta}$ as a vector function of the *m* source parameters $\theta_1^s,\ldots,\theta_m^s$ that are, according to our assumption, identical to the first *m* components of $\boldsymbol{\theta}$. The vector function implements the functional constraints on parameters and may be written as a system of *n* equations:

$$\begin{aligned}
\theta_1 &= \theta_1^s \\
\theta_2 &= \theta_2^s \\
&\vdots \\
\theta_m &= \theta_m^s \\
\theta_{m+1} &= f_{m+1}\left(\theta_1^s,\ldots,\theta_m^s\right) \\
\theta_{m+2} &= f_{m+2}\left(\theta_1^s,\ldots,\theta_m^s\right) \\
&\vdots \\
\theta_n &= f_n\left(\theta_1^s,\ldots,\theta_m^s\right)
\end{aligned}.$$

The Jacobian matrix **J** is the $n\times m$ matrix of partial derivatives of the components of the vector $\boldsymbol{\theta}$ with respect to the components of the vector $\boldsymbol{\theta}^s$ of source parameters:

$$\mathbf{J} = \begin{bmatrix}
\frac{\partial\theta_1}{\partial\theta_1^s} & \cdots & \frac{\partial\theta_1}{\partial\theta_m^s} \\
\vdots & & \vdots \\
\frac{\partial\theta_m}{\partial\theta_1^s} & \cdots & \frac{\partial\theta_m}{\partial\theta_m^s} \\
\frac{\partial\theta_{m+1}}{\partial\theta_1^s} & \cdots & \frac{\partial\theta_{m+1}}{\partial\theta_m^s} \\
\vdots & & \vdots \\
\frac{\partial\theta_n}{\partial\theta_1^s} & \cdots & \frac{\partial\theta_n}{\partial\theta_m^s}
\end{bmatrix} = \begin{bmatrix}
1 & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & 1 \\
\frac{\partial\theta_{m+1}}{\partial\theta_1^s} & \cdots & \frac{\partial\theta_{m+1}}{\partial\theta_m^s} \\
\vdots & & \vdots \\
\frac{\partial\theta_n}{\partial\theta_1^s} & \cdots & \frac{\partial\theta_n}{\partial\theta_m^s}
\end{bmatrix},$$

where $\frac{\partial\theta_j}{\partial\theta_i^s} = \frac{\partial f_j\left(\theta_1^s,\ldots,\theta_m^s\right)}{\partial\theta_i^s}$ denotes the partial derivative of the functional constraints on parameter $\theta_j = f_j\left(\theta_1^s,\ldots,\theta_m^s\right)$ with respect to the source parameter θ_i^s.

Note that, due to the first functional constraints being equality constraints, the first $m\times m$ submatrix of **J** is the $m\times m$ identity matrix.

The computation of standardized residuals in the presence of functional constraints requires the specification of a function that takes as input the (full) parameter vector and computes that *full* $n\times n$ Jacobian matrix, i.e., the Jacobian matrix that also contains the columns representing parameters subject to functional constraints. The full Jacobian matrix looks like this.

$$\mathbf{J}_{\text{full}} = \begin{bmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 1 & 0 & \cdots & 0 \\ \frac{\partial\theta_{m+1}}{\partial\theta_1^s} & \cdots & \frac{\partial\theta_{m+1}}{\partial\theta_m^s} & 1 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\theta_n}{\partial\theta_1^s} & \cdots & \frac{\partial\theta_n}{\partial\theta_m^s} & 0 & \cdots & 1 \end{bmatrix}.$$

The gradient vector $\nabla_{\boldsymbol{\theta}^s}$ of partial derivatives of the log likelihood function with respect to the source parameters is given by pre-multiplying **J** with the gradient vector $\nabla_{\boldsymbol{\theta}}$:

$$\nabla_{\boldsymbol{\theta}^s} = \nabla_{\boldsymbol{\theta}}^{\mathbf{T}} \cdot \mathbf{J}.$$

Component *j* of the new gradient vector representing the partial derivative of the log likelihood function with respect to source parameter *j* is thus given by:

$$\begin{aligned} \frac{\partial \log L}{\partial\theta_j^s} &= \frac{\partial \log L}{\partial\theta_1}\cdot\frac{\partial\theta_1}{\partial\theta_j^s} + \frac{\partial \log L}{\partial\theta_2}\cdot\frac{\partial\theta_2}{\partial\theta_j^s} + \cdots + \frac{\partial \log L}{\partial\theta_n}\cdot\frac{\partial\theta_n}{\partial\theta_j^s} \\ &= \frac{\partial \log L}{\partial\theta_1}\cdot\underbrace{\frac{\partial\theta_1}{\partial\theta_j^s}}_{0} + \cdots + \frac{\partial \log L}{\partial\theta_j}\cdot\underbrace{\frac{\partial\theta_j}{\partial\theta_j^s}}_{1} + \cdots + \frac{\partial \log L}{\partial\theta_j}\cdot\underbrace{\frac{\partial\theta_j}{\partial\theta_m^s}}_{0} + \frac{\partial \log L}{\partial\theta_{m+1}}\cdot\frac{\partial\theta_{m+1}}{\partial\theta_m^s} + \cdots + \frac{\partial \log L}{\partial\theta_n}\cdot\frac{\partial\theta_n}{\partial\theta_m^s}. \\ &= \frac{\partial \log L}{\partial\theta_j} + \frac{\partial \log L}{\partial\theta_{m+1}}\cdot\frac{\partial\theta_{m+1}}{\partial\theta_m^s} + \cdots + \frac{\partial \log L}{\partial\theta_n}\cdot\frac{\partial\theta_n}{\partial\theta_m^s} \end{aligned}$$

The matrix multiplication implements the chain rule. However, the outlined procedure for computing the adjustments of the parameter and gradient vector is computationally inefficient, for two reasons:

1. No modification of parameters that function as source parameters is required. Functional constraints have to be specified for the target parameters only.
2. In case of few functional constraints **J** consists predominately of zeros resulting in the summation of many zero components.

To circumvent these problems, the specification of functional constraints as well as the adjustment of the gradient vector is performed by means of a user defined function that performs two actions:

1. The functional constraints on the target parameters are specified , i.e., the target parameters are specified as functions of the source parameters. No modification of source parameters is performed.
2. The entries of gradient parameter associated with the source parameters are adjusted only. The other entries of the gradient vector are left unchanged.

The function has the following general structure (the "…" indicates the code that has to be specified by the user:

```
functional <- function(par)
{
      . . .             # Specification of functional constraints

      gr.fct <- function(grad, par)
      {
      . . .             # Adjustment of relevant components of the gradient
      return(grad)
      }
      attr(par, "SDT.gradient") <- gr.fct # Gradient function is passed as
                                          # an attribute of the parameter

      return(par)
}
```

Note that the (user defined) gradient function has to be passed as an attribute named "SDT.gradient" to the parameter vector (for concrete examples, cf., Section 4.4.2).

In addition to the specification of the function for implementing functional constraints, the target parameters subject to functional constraints must be specified as *fixed parameters*, i.e., they must be added to the matrix of fixed constraints. This guarantees that these parameters as well as their partial derivatives are eliminated from the parameter and gradient vector, respectively.

In case of employment of the analytically computed model matrix (e.g. for the computation of standardized residuals [cf. Chapter 2.2]), a function computing the Jacobian matrix corresponding to the functional constraints has to be defined and passed with the attribute `"SDT.Jacobian"` to the function (computing the functional constraints):

```
J.fct <- function(par)
{
. . .           # Specification of Jacobian matrix J
return(J)
}
attr(functional, "SDT.Jacobian") <- J.fct
```

The parameters that are the targets of the constraints make up the rows of the Jacobian matrix (i.e. positions of target parameters within the parameter vector are the row indices) whereas the positions of source parameters of the constraints make up the columns of the Jacobian matrix.

4.4.2 Examples

In order to concretize the preceding considerations a number of examples are provided. The first example illustrates the structure of the Jacobian matrix.

Example 1:

Let $\theta_3 = 3 \cdot \theta_1^{s^2} \cdot \theta_2^s$, then $\frac{\partial \theta_3}{\partial \theta_1^s} = 6 \cdot \theta_1^s \cdot \theta_2^s$ and $\frac{\partial \theta_3}{\partial \theta_2^s} = 3 \cdot \theta_1^{s^2}$; the resulting Jacobian is

$$\mathbf{J} = \begin{bmatrix} \frac{\partial\theta_1^s}{\partial\theta_1^s} & \frac{\partial\theta_1^s}{\partial\theta_2^s} \\ \frac{\partial\theta_2^s}{\partial\theta_1^s} & \frac{\partial\theta_2^s}{\partial\theta_2^s} \\ \frac{\partial\theta_3}{\partial\theta_1^s} & \frac{\partial\theta_3}{\partial\theta_2^s} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 6\cdot\theta_1^s\cdot\theta_2^s & 3\cdot\theta_1^{s^2} \end{bmatrix},$$

and the full Jacobian is:

$$\mathbf{J}_{\text{full}} = \begin{bmatrix} \frac{\partial\theta_1^s}{\partial\theta_1^s} & \frac{\partial\theta_1^s}{\partial\theta_2^s} & \frac{\partial\theta_1^s}{\partial\theta_3} \\ \frac{\partial\theta_2^s}{\partial\theta_1^s} & \frac{\partial\theta_2^s}{\partial\theta_2^s} & \frac{\partial\theta_2^s}{\partial\theta_3} \\ \frac{\partial\theta_3}{\partial\theta_1^s} & \frac{\partial\theta_3}{\partial\theta_2^s} & \frac{\partial\theta_3}{\partial\theta_3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 6\cdot\theta_1^s\cdot\theta_2^s & 3\cdot\theta_1^{s^2} & 1 \end{bmatrix},$$

The next example demonstrates the implementation of functional constraints with the general Gaussian signal detection model (cf. Chapter 3.2).

Example 2:
In some applications it is required to set the value of one parameter equal to minus the value of another parameter (e.g., the mean of one distribution should be set to minus the mean of another distribution). This type of constraint can easily be implemented by means of functional constraints.
Assume that we have an SDT model with two distributions (for two types of signals, e.g., noise and noise+signal). For each signal, a location parameter (`m`), a scale parameter (`std`), and threshold parameters (`ti`) are specified (i.e., the general Gaussian model is used). The vector of starting parameter may look like this:

```
#           m  std    t1    t2   t3   t4   t5
par <- c(0.0, 1.0, -1.0, -0.5, 0.5, 1.0, 1.7,   # Signal 1
         0.0, 1.0, -1.0, -0.5, 0.5, 1.0, 1.7,   # Signal 2
```

Instead of fixing the mean of the first distribution to a specific value, we impose the constraint that the mean of the first distribution is equal to minus the mean of the second distribution: `m1 = -m2`.
The subsequent function `fct.m` performs the adjustment of the parameters and the embedded function `gr.fct` performs the adjustment of the gradient.

The function `J.fct` following to `fct.m` computes the full Jacobian matrix. The function `J.fct` is attached to that main function `fct.m` as the attribute `"SDT.Jacobian"`.

```
# FUNCTION PERFOMING FUNCTIONAL CONSTRAINT: m1 = -m2
fct.m <- function(par)
{
      m1 <- 1           # Target index: mean 1 on position 1
      m2 <- 8           # Source index: mean 2 on position 8

      par[m1] <- -par[m2]

      # Define Gradient function
      gr.fct <- function(grad, par)
      {
            grad[m2] <- grad[m2] - grad[m1]
            return(grad)
      }
      attr(par, "SDT.gradient") <- gr.fct # Gradient
      return(par)
}

# FUNCTION COMPUTING THE CORRESPONDING JACOBIAN MATRIX
J.fct <- function(par)
{
      J <- diag(length(par))
      J[1, 8] <- -1
      J
}
# PASSING FUNCTION FOR COMPUTING THE JACOBIAN MATRIX AS AN ATTRIBUTE
# "SDT.Jacobian" TO THE FUNCTION
attr(fct.m, "SDT.Jacobian") <- J.fct
```

The specification of the function `fct.m` comprises the following parts:

1. The target parameter `m1` is located in Position 1 within the parameter vector, and the source parameter `m2` is locate at the 8. position.
2. The command `par[m1] <- -par[m2]` specifies the functional constraint.
3. The gradient function `gr.fct` is specified, with two parameters as input: `grad`, the gradient vector, and `par`, the parameter vector.
4. The new value of the derivative of the log likelihood function with respect to the source parameter `m2` is computed. It is simply the original derivative (located in `grad[m2]`) minus the negative of the derivative with respect to the target index (`-grad[m1]`). Only this component of the gradient vector has to be adjusted.
5. The new gradient with the indicated modification is returned.
6. The next command `attr(par, "SDT.gradient") <- gr.fct` specifies the gradient function as an attribute of the parameter vector. As a result, the function is passed together with the parameter vector and can be used by the internal function that actually computes the gradient. This function simply applies `gr.fct()` to perform the modification of the gradient due to the functional constraint.

The specification of the function `J.fct`, computing the full Jacobian matrix, consists in the following parts:

1. An identity matrix with the number of parameters (including free and constrained ones) is generated. In the present case this is a 14×14 matrix with 1 in the main diagonal and 0 elsewhere.
2. The entry [1,8] of matrix **J** representing the partial derivative $\partial m_1 / \partial m_2 = -1$ is specified. Note that the position of the target parameter (= 1) corresponds to the row index whereas the position of the source parameter (= 8) corresponds to the column index.

Comment: This type of constraint is used in the example in Chapter 7.1.

The next example demonstrates the implementation of slightly more complex functional constraints within the standard SDT model with the distribution N(0, 1) representing the first signal (cf. Chapter 3.1).

Example 3:

Assume the SDT model with two signals, and with the first signal represented by the standard normal distribution N(0, 1). The parameter vector might look like this:

```
#         m2  std2    t1    t2   t3   t4   t5
par <- c(1.0,  1.0, -1.0, -0.5, 0.5, 1.0, 1.7)
```

`m2` and `std2` denote, respectively, the mean and the standard deviation of the distribution representing the second signal, whereas `t1 – t5` denote the decision bounds (assuming six response categories).

We would like to implement the following symmetry restrictions on the decision bounds:

$$\begin{aligned} t_3 - t_2 &= t_4 - t_3 \\ t_3 - t_1 &= t_5 - t_3 \end{aligned},$$

that is, we assume that decision bounds are spaced symmetrical around t_3.
Taking $t_1 - t_3$ as source and t_4 and t_5 as target parameters the restrictions can be reformulated:

$$\begin{aligned} t_4 &= 2 \cdot t_3 - t_2 \\ t_5 &= 2 \cdot t_3 - t_1 \end{aligned}.$$

The following piece of code performs the relevant computations:

```
# FUNCTION PERFOMING FUNCTIONAL CONSTRAINT ON THRESHOLDS:
#     t3 - t2 = t4 - t3
#     t3 - t1 = t5 - t3
fct.m <- function(par)
{
      t.1 <- 3          # Position of decision bound 1
      t.2 <- 4          # Position of decision bound 2
      t.3 <- 5          # Position of decision bound 3
      t.4 <- 6          # Position of decision bound 4
      t.5 <- 7          # Position of decision bound 5

      par[t.4] <- 2*par[t.3] - par[t.2]
      par[t.5] <- 2*par[t.3] - par[t.1]


      # Define Gradient function
      gr.fct <- function(grad, par)
      {
            grad[t.3] <- grad[t.3] + 2*grad[t.4] + 2*grad[t.5]
            grad[t.2] <- grad[t.2] - grad[t.4]
            grad[t.1] <- grad[t.1] - grad[t.5]
            return(grad)
      }
      attr(par, "SDT.gradient") <- gr.fct             # Gradient

      return(par)
}
```

```
# FUNCTION COMPUTING THE CORRESPONDING JACOBIAN MATRIX
J.fct <- function(par)
{
      J <- diag(length(par))
      J[6, 5] <-  2
      J[7, 5] <-  2
      J[6, 4] <- -1
      J[7, 3] <- -1
      J
}
# PASSING FUNCTION FOR COMPUTING THE JACOBIAN MATRIX AS AN ATTRIBUTE
# "Jacobian" TO THE FUNCTION
attr(fct.m, "SDT.Jacobian") <- J.fct
```

In the previous two examples the adaptation of the gradient as well as the computation of the Jacobian matrix does not require the actual parameter vector. This is due to the fact that linear constraints were implemented only. However, the method can be used to implement arbitrary linear and non-linear constraints on parameters. The following, though somewhat artificial, example demonstrates this.

Example 4:

Assume that the parameter vector $\boldsymbol{\theta}$ contains four components: $\boldsymbol{\theta} = \left[\theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4\right]^{\mathbf{T}}$, and the user would like to specify the following functional constraints:

$$\begin{aligned} \theta_3 &= 5 \cdot \theta_1^2 + \log \theta_2 \\ \theta_4 &= \theta_2^3 \end{aligned} .$$

Thus, parameters θ_1 and θ_2 are the source parameters whereas parameters θ_3 and θ_4 are the target parameters. By consequence, the Jacobian matrix **J** is 4×2 matrix and the full Jacobian matrix $\mathbf{J}_{\text{full}}$ 4×4 matrix:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial\theta_1}{\partial\theta_1} & \frac{\partial\theta_1}{\partial\theta_2} \\ \frac{\partial\theta_2}{\partial\theta_1} & \frac{\partial\theta_2}{\partial\theta_2} \\ \frac{\partial\theta_3}{\partial\theta_1} & \frac{\partial\theta_3}{\partial\theta_2} \\ \frac{\partial\theta_4}{\partial\theta_1} & \frac{\partial\theta_4}{\partial\theta_1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 10 \cdot \theta_1 & \frac{1}{\theta_2} \\ 0 & 3 \cdot \theta_2^2 \end{bmatrix} \qquad \mathbf{J}_{\text{full}} = \begin{bmatrix} \frac{\partial\theta_1}{\partial\theta_1} & \frac{\partial\theta_1}{\partial\theta_2} & \frac{\partial\theta_1}{\partial\theta_3} & \frac{\partial\theta_1}{\partial\theta_4} \\ \frac{\partial\theta_2}{\partial\theta_1} & \frac{\partial\theta_2}{\partial\theta_2} & \frac{\partial\theta_2}{\partial\theta_3} & \frac{\partial\theta_2}{\partial\theta_4} \\ \frac{\partial\theta_3}{\partial\theta_1} & \frac{\partial\theta_3}{\partial\theta_2} & \frac{\partial\theta_3}{\partial\theta_3} & \frac{\partial\theta_3}{\partial\theta_4} \\ \frac{\partial\theta_4}{\partial\theta_1} & \frac{\partial\theta_4}{\partial\theta_2} & \frac{\partial\theta_4}{\partial\theta_3} & \frac{\partial\theta_4}{\partial\theta_4} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 10 \cdot \theta_1 & \frac{1}{\theta_2} & 1 & 0 \\ 0 & 3 \cdot \theta_2^2 & 0 & 1 \end{bmatrix}$$

The new gradient vector $\nabla_{\boldsymbol{\theta}^s} = \nabla_{\boldsymbol{\theta}}^{\mathbf{T}} \cdot \mathbf{J}$ consists of the following entries:

$$\begin{bmatrix} \frac{\partial L}{\partial \theta_1^s} \\ \frac{\partial L}{\partial \theta_2^s} \end{bmatrix}^{\mathbf{T}} = \begin{bmatrix} \frac{\partial L}{\partial \theta_1} + 10 \cdot \theta_1 \cdot \frac{\partial L}{\partial \theta_3} \\ \frac{\partial L}{\partial \theta_2} + \frac{1}{\theta_2} \cdot \frac{\partial L}{\partial \theta_3} + 3 \cdot \theta_2^2 \cdot \frac{\partial L}{\partial \theta_4} \end{bmatrix}^{\mathbf{T}}$$

The function for specifying these functional constraints and for the adjustment of the gradient vector looks like this:

```
functional <- function(par)
{
      par[3] <- 5*par[1]^2 + log(par[2])
      par[4] <- par[2]^3

      gr.fct <- function(grad, par)
      {

            grad[1] <- grad[1] + 10*par[1]*grad[3]
            grad[2] <- grad[2] + 1/par[2]*grad[3] + 3*par[2]^2*grad[4]
            return(grad)
      }
      attr(par, "SDT.gradient") <- gr.fct # Gradient function is passed as
                                          # an attribute of the parameter

      return(par)
}
```

The function for computing the full Jacobian matrix looks like this:

```
J.fct <- function(par)
{
      J <- diag(4)
      J[3, 1] <- 10*par[1]
      J[3, 2] <- 1/par[2]
      J[4, 2] <- 3*par[2]^2
      J
}
# PASSING FUNCTION FOR COMPUTING THE JACOBIAN MATRIX AS AN ATTRIBUTE
# "SDT.Jacobian" TO THE FUNCTION
attr(functional, "SDT.Jacobian") <- J.fct
```

4.4.3 Concluding remarks

(i) The functional constraints are of highest priority, that is, they override possible other constraints (fixed or equality constraints specified by means of the methods described above). As a result the value specified for the fixed constraint is arbitrary.

(ii) The user-defined function is passed to the estimation function `SDT.Estimate()` in the argument `functional`.

(iii) This method can also be used for specifying fixed and equality constraints. However, it is preferable to specify these types of constraints by means of matrices, as detailed above. This exempts the user from specifying components of gradients as well as Jacobians.

(iv) It is possible to define functional constraints without specifying an associated gradient function. In this case, the argument `sym.gr` of `SDT.Estimate()` (cf. above, Section 2.1) must be set to `sym.gr = F`. By consequence, the gradient is approximated numerically. This approximation retards the process of estimation considerably. The effect on the precision of the estimates is ignorable, however. In addition, the Jacobian matrix is also computed numerically at the end of the process of estimation. For none of the examples a relevant difference between analytically and numerically computed Jacobian matrices was observed.

Warning:

A parameter subjected to functional constraints must not be involved in an equality constraint neither as the source nor as the target. This is due to the fact that the functional constraints are set after equality constraints. As result, an equality constraint involving the parameter (as a target) is not set correctly because it does not incorporate the functional constraint (The usage of the parameter as a target of an equality constraint results in a faulty computation of the gradient).
Consequently, in case of parameters being subject to equality as well as to more complex functional constraints, the equality constraints must be specified within the function for defining the functional constraints.

Comment: Additional examples demonstrating the specification of complex functional constraints and the respective adjustment of the gradient vector, as well as the computation of the Jacobian matrix are presented in the following chapters.

5. Working Examples: SDT and Gaussian model

In the following, different examples for fitting SDT models with the SDT and the Gaussian version of the SDT model are demonstrated. We start with a simple example where the SDT model is fitted to the data with two types of signal (new vs old) and two response categories per signal: “new” vs. “old”. For this first example the complete R code, including all comments, is discussed in order to provide a thorough explanation of the model functions.
This is followed by a demonstration of how to fit the same data with the Gaussian version of the model assuming a *symmetric configuration*, i.e., the mean of the noise distribution being minus the mean of the signal distribution. The further examples reveal additional facilities of the module.

5.1 Example 1: Fitting the SDT model to data involving two types of signals

In the simplest case, data from two types of signals (noise vs. signal+noise or new vs. old) with two response categories per signal are available (“yes” vs. “no” or “new” vs. “old” etc.). The first part demonstrates how to fit these type of data using the standard SDT model whereas the second part demonstrates the usage of the Gaussian version of the model using a symmetric configuration.

5.1.1 Example 1-1: Fitting the standard SDT model to Yes/No data

The R code shown below is contained in the example file: `SDT-1 (Y-N-Data).R`.

```
# ================================================================
# Examples of the documentation of SDT Models:
# SDT-1: Modeling of Yes/No data
#
# Date of last update: August, 2013
# Autor: Siegfried Macho
# ================================================================

library(numDeriv)

# DIRECTORIES (PLEASE ADJUST TO YOUR PERSONAL SETTING)
DirSource <- "C:/Documents and Settings/macho/Statistik/_SDT-Models
(Version 2)/R-Code/"
DirSourceM <- "C:/Documents and Settings/macho/Statistik/_SDT-Models
(Version 2)/R-Code/Models/"
```

```
# FILE NAME WITH THE MODEL
Model <- "SDT-SDT.R"

# FILE NAMES FOR ESTIMATION
Main      <- "SDT-Main.R"
Auxiliary <- "SDT-Auxiliary.R"

# LOAD SOURCE FILES WITH SDT MODEL
source(paste(DirSourceM, Model, sep = ""))

# LOAD SOURCE FILES WITH AUXILIARY AND ESTIMATION FUNCTIONS
source(paste(DirSource, Auxiliary, sep = ""))
source(paste(DirSource, Main, sep = ""))


datavec <- c(1780, 763,       # NEW
             883, 1025)       # OLD

# List with configuration information: Both distributions are N(0, 1)
cfg   <- list(n.sdt = 2, restriction = "equalvar")

# Print Parameters
PI <- SDT.Parameter.Info(data = datavec, n = cfg)
cat("\n-----------------------------------------\n")
cat("  Parameter-Information (SDT-Model):")
cat("\n-----------------------------------------\n")
print(PI)

# ESTIMATE AND EVALUATE THE RESULTS
Opti.Obj <- SDT.Estimate(data = datavec, n = cfg, test = T)
Stat.Obj <- SDT.Statistics(Opti.Obj, control = list(LR = T))
print(Stat.Obj)

# PLOT CONFIGURATION
SDT.Plot(Opti.Obj, cols = 3:4, labels = c("Memory Strength", "Density"),
SDT.legend = list(text = c("New", "Old")))
```

This piece of R-code performs the following steps:

1. The library `NumDeriv` containing functions for computing highly accurate numerical derivatives (Gilbert, 2006) is loaded:

```
library(numDeriv)
```

2. The name of the directory containing the main files as well as the directory containing the model files are specified:

```
DirSource <- "C:/Documents and Settings/macho/Statistik/_SDT-Models
(Version 2)/R-Code/"
DirSourceM <- "C:/Documents and Settings/macho/Statistik/_SDT-Models
(Version 2)/R-Code/Models/"
```

3. The name of file containing the SDT model is specified.

```
Model <- "SDT-SDT.R"
```

4. The names of the source files with the estimation routines are specified:

```
Main      <- "SDT-Main.R"
Auxiliary <- "SDT-Auxiliary.R"
```

5. The model is loaded:

```
source(paste(DirSourceM, Model, sep = ""))
```

Comment: The `paste()` function is used to concatenate the directory and the file name.

6. The two main files are loaded:

```
source(paste(DirSource, Auxiliary, sep = ""))
source(paste(DirSource, Main, sep = ""))
```

7. The data vector with response frequencies for the two stimulus sets (corresponding to the two types of signals) is specified:

```
datavec <- c(1780, 763,       # NEW
             883, 1025)       # OLD
```

The first number (1780) is the number of "new" responses to new items, the second number (763) is number of "old" responses to new items, the third number represents the number of "new" reponses to old items, and, finally, the last number (1025) is the number of "old" responses to old items.

8. The list of configuration information is specified:

```
cfg   <- list(n.sdt = 2, restriction = "equalvar")
```

The first entry `n.sdt = 2` indicates that there are two types of signal or, equivalently, two Gaussian distributions. The second entry `restriction = "equalvar"` tells the program to set the variance parameter of the Gaussian distribution, representing the old items, to 1.0.

Comments:

- The restriction of equal variances of the two distributions is required for the identification of the model.
- The first entry of the list is not required since `n.sdt = 2` is the default that is supplied by the program in case of no specification being provided by the user.

9. The Parameter information is calculated and stored in the variable `PI`.

```
PI <- SDT.Parameter.Info(data = datavec, n = cfg)
```

Comment:

For more information on the function `SDT.Parameter.Info`, cf. Chapter 2.4.

10. The parameter information is printed (together with a header):

```
cat("\n-----------------------------------------\n")
cat("  Parameter-Information (SDT-Model):")
cat("\n-----------------------------------------\n")
print(PI)
```

11. The estimation function is called:

```
Opti.Obj <- SDT.Estimate(data = datavec, n = cfg, test = T)
```

The function `SDT.Estimate` requires at least one argument: the vector with the data has to be passed to the function with the argument `data`. The function returns the optimization object `Opti.Obj` (The structur of the optimization object is described in Chapter 2.1)

Comment:

The name of the argument `data` must be specified because for the first argument (`par`) of the function the default value is used. If the argument name were not specified the function would interpret the input as the parameter vector (For a full description of the estimation function, cf. Chapter 2.1).

In the present case, the list with configuration information is passed as the second argument: `n = cfg`.
Finally, the test flag `test` is set to `TRUE`. By consequence, the gradient is computed numerically at the optimum using the function `grad()` from the package `numDeriv` and compared to the analytically computed gradient. The result is printed in the output.

Comment:
It is good practice to set `test = TRUE` because non-negligible differences between symbolic and numeric gradients indicate problems of the estimation process, for example, if an estimate is on the border of the parameter space (e.g. a standard deviation parameter near zero).

12. The result of the estimation is evaluated:

```
Stat.Obj <- SDT.Statistics(Opti.Obj, control = list(LR = T))
```

The function `SDT.Statistics()` usually requires a single argument, the optimization object that is computed by the function `SDT.Estimate()`. In the present case a second argument is passed to the function that instructs the function to compute likelihood ratio confidence intervals for the estimated mean (*d*').
The result of the computation is stored in the statistics object `Stat.Obj`.

Comment:
A full documentation of the function `SDT.Statistics()` may be found in Chapter 2.2.

13. The content of the statistics object `Stat.Obj` is displayed:

```
print(Stat.Obj)
```

14. The configuration is plotted:

```
SDT.Plot(Opti.Obj, cols = 3:4, labels = c("Memory Strength", "Density"),
SDT.legend = list(text = c("New", "Old")))
```

The function `SDT.Plot()` also takes the information from the optimization object. In addition. The argument `cols = 3:4` indicates that the colors blue and red should be used for the density curves (cf. Figure 11). Furthermore, labels for the *x*- and *y*- axes are provided as well as a legend.

Comment:
A full documentation of the function `SDT.Plot()` may be found in Chapter 2.5.

The R code for estimating the model and printing the test statistics seems to be quite complicated. However, the main portion consists essential of 4 components:

1. Specification of the data:

```
datavec <- c(1780, 763,       # NEW
             883, 1025)       # OLD
```

2. Specification of the configuration information:

```
cfg   <- list(n.sdt = 2, restriction = "equalvar")
```

3. Estimation of parameters:

```
Opti.Obj <- SDT.Estimate(data = datavec, n = cfg, test = T)
```

4. Evaluation and printing of the estimation results:

```
Stat.Obj <- SDT.Statistics(Opti.Obj, control = list(LR = T))
print(Stat.Obj)
```

The printed output consists of two parts:

(a) The parameter information, and
(b) The results of the estimation.

The following parameter information is printed:

```
$Model
[1] "Standard SDT model with noise distribution N(0, 1), Number of models:
<2>, Type of restrictions: <equalvar>"

$Parameters.and.Constraints
       name  par fixed.value ident.source Nr
1   Mean[2]  0.0         ---          ---
2 Stddev[2]  1.0     1 <set>          ---
3       t-1 -0.9         ---          ---
```

The printed parameter information comprises two sections:

(i) The first section called `$Model` contains a string that provides information about the model and its configuration. In the present case the string tells us that the SDT model is used with new distribution identical to *N*(0, 1) that the number of signals is 2 and that equal variance restrictions are set.

(ii) The second section headed `$Parameters.and.Constraints` contains the names of the parameters their starting values and which of them were fixed or set equal. In the present case the model comprises three parameters: the mean `Mean[2]` and the standard deviation `Stddev[2]` of the old distribution, as well as a threshold parameter `t-1` (The parameters of the new distribution are not shown because they are always equal to zero and one). The standard deviation of the old distribution is fixed to 1.0 as indicated by the marker `<set>`.

The printed results of the estimation look like this:

```
=====================================================
 Symbolic and numeric gradients at the optimum:
 Length of difference vector  : 1.193788e-08
 Length of gradient at optimum: 8.290594e-06
=====================================================
         Symbolic   Numeric Difference
Mean[2]  4.27e-06  4.28e-06     -1e-08
t-1     -7.11e-06 -7.11e-06      0e+00
=====================================================
$Model.description
[1] "Standard SDT model with noise distribution N(0, 1), Number of models: <2>,
Type of restrictions: <equalvar>"

$Statistics
                                        Statistic
log L                                   -2870.749
X^2                                         0.000
G^2                                         0.000
df                                          0.000
p(Y > X^2)                                     NA
p(Y > G^2)                                     NA
AIC                                      5745.497
BIC                                      5758.299
CAICF                                    5776.687
ICOMP                                    5742.253
ICOMP.R                                  5742.100
Free Parameters                             2.000
Length of gradient at optimum               0.000
Rank of Hessian                             2.000
Condition number of information matrix      6.360
Rank of model matrix: t(J) * J              2.000
Condition number of model matrix            7.859
```

```
$Free.parameters
        Value    SE CFI-95(Lower) CFI-95(Upper) LR-95(Lower) LR-95(Upper)
Mean[2] 0.618 0.039         0.542         0.694        0.542        0.694
t-1     0.524 0.026         0.473         0.576

$Full.parametervector
       Gauss-1 Gauss-2
Mean     0.000   0.618
Stddev   1.000   1.000
t-1      0.524   0.524

$SDT.measures
       Value    SE CFI-95(Lower) CFI-95(Upper)
d.a[2] 0.618 0.039         0.542         0.694
d.e[2] 0.618 0.039         0.542         0.694
A.z[2] 0.669 0.010         0.649         0.688

$Data.and.Estimates
             Observed Expected Probabilities Residuals  Std.Resid.(analytical)
Signal 1 [1]     1780     1780         0.700         0                       0
Signal 1 [2]      763      763         0.300         0                       0
Signal 2 [1]      883      883         0.463         0                       0
Signal 2 [2]     1025     1025         0.537         0                       0
```

The table at the beginning exhibits the difference between the analytically and the numerically computed gradient at the optimum (The difference shown is the square root of the sum of the squared differences of the single components of the two vectors). In the present case, the difference between the two is negligible. The table further indicates that the gradient at the optimum is close to 0.0, as it is necessary for the estimation to be successful.
The rows of the table contain the individual components of the symbolic and the numeric gradient vector as well as their difference. Each row corresponds to a free parameter. Again, all entries should be close to zero.
This table is printed only if the flag `test` of the estimation function is set to `TRUE`.

The residual output presents the content of the statistics object `Stat.Obj` that is computed by the procedure `SDT.Statistics()`. The statistics object contains 5 sections:

1. The section `$Model.description` contains a string indicating the model actually used (This string is identical to that provided above with the parameter information).
2. The section `$Statistics` contains various fit statistics, like Pearson `X^2` and Likelihood ratio statistic `G^2`, the degrees of freedom `df` associated with these statistics, as well as other statistics that are relevant for evaluating the quality of the fit, like, for example, the rank and the condition number of the observed information matrix.
3. The section `$Free.parameters` contains the names and values of the estimated parameters and the estimated standard errors (SE) as well as the limits of the confidence intervals. Due to the fact that likelihood ratio intervals were computed, two conficence intervals are shown for the mean, the first interval is based on the assumption that estimated parameters are normally distributed with the estimated standard error. The likelihood ratio confidence intervals do not require the normality assumption and are thus more reliable. In the present case both types of intervals are identical [A good description of likelihood ratio confidence intervals may be found in Pawitan (2001)].
4. The section `$Full.parametervector` contains the full parameter vector including also those parameters that are subjected to constraints or that are added internally by the program.

5. The section `$SDT.measures` contains the estimated values, standard errors and confidence intervals for a number of important signal detection measures, like A_z, the area under the estimated ROC curve, and d_a, the Mahalanobis distance between the two distributions. With unequal variances these measures may be better indicators of sensitivity as *d'* (for further details, cf. Section 2.2).
6. The section `$Data.and.Estimates` contains the data in the order provided by the input as well as estimated frequencies, probabilities, the Pearson residuals as well as the standardized residuals that are asymptotically standard normal distributed. The row labels `Signal s[r]` indicate the signal (`s`) and the response (`r`) within the signal.

(For further details concerning the contents of the different sections in the output, cf. Chapter 2.2).

The Figure 11 shows the positions of the estimated density curves as well as of the threshold (the grey vertical line). The figure was generated by the function `SDT.Plot()`.

Following to this detailed description of the fitting of the SDT model, the fitting of symmetric setup of the distributions with $\mu_1 = -\mu_2$ is illustrated. This model can be fitted only with the Gaussian version of the SDT model.

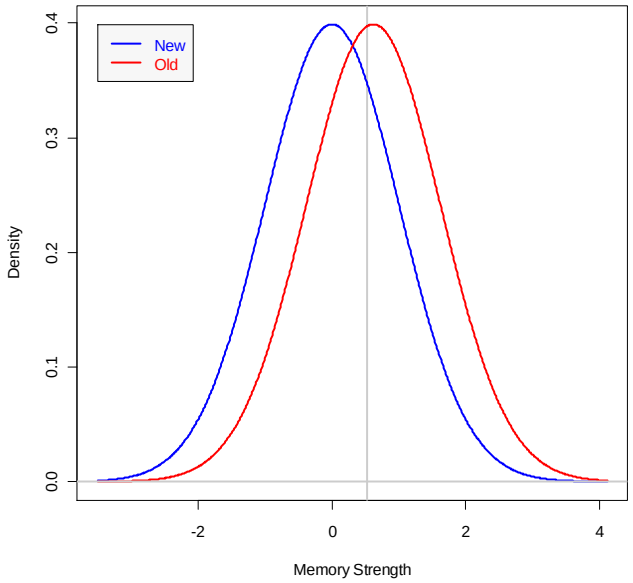


Figure 11: *Density functions plotted by the function* `SDT.Plot()`.

5.1.2 Example 1-2: Fitting the symmetric SDT model to Yes/No data

The R code for his example is contained in the example file:

`Gauss-1.2 (Y-N-Data, symmetric).R`.

In the following, I present only the relevant differences to the above example. The R code for this examples differs from the previous one in two respects only:

1. The list of configuration information has been modified:

```
cfg <- list(n.sdt = 2, restriction = "equalvar-symmetric")
```

The string passed to the option `restriction` now indicates that an equal variance model with $\mu_1 = -\mu_2$ should be fitted. This type of restriction is valid only for the Gaussian model and not for the standard SDT model.

2. An additional argument is passed to the estimation function:

```
Opti.Obj <- SDT.Estimate(data = datavec, n = cfg, Model.Id = "Gaussian",
test = T)
```

The argument `Model.Id = "Gaussian"` indicates that the Gaussian model should be fitted to the data instead of the standard SDT model.

Comment:
For the standard SDT model the model identification string passed to the argument `Model.Id` is `"SDT"`. Due to the fact that this is the default option no model identification string is required for the standard SDT model.

Here is an excerpt of the relevant portion of the output showing the estimated free parameters with estimated standard errors and confidence intervals (likelihood ratio confidence intervals have not been not computed):

```
[...]
$Free.parameters
        Value    SE CFI-95(Lower) CFI-95(Upper)
t-1[1]  0.215 0.019         0.177         0.254
Mean[2] 0.309 0.019         0.271         0.347

$Full.parametervector
       Gauss-1 Gauss-2
Mean    -0.309   0.309
Stddev   1.000   1.000
t-1      0.215   0.215
[...]
```

The output confirms that $\mu_1 = -\mu_2$ with estimated $d' = 0.618$ being identical to that estimated by the standard SDT model. Clearly, the position of the threshold is now different from that estimated by the standard SDT model.

5.2 Example 2: Fitting rating data

The present example illustrates the fitting of rating data using a data set of Ratcliff, McKoon and Tindall (1994, Experiment 1, p. 783). The R-code of this example may be found in the file: `SDT-2.2 (2 signals, Data Ratcliff).R`

Here is the relevant piece of code for fitting the data, printing the results and plotting densitiy and ROC curves:

```
data <- c(477, 776, 527, 321, 258, 184,   # New
          192, 401, 290, 267, 316, 442)   # Old
Opti.Obj <- SDT.Estimate(data = data, test = T)
Stat.Obj <- SDT.Statistics(Opti.Obj)      # Evaluate the results
print(Stat.Obj)
# PLOT CONFIGURATION
# Plot density curves
SDT.Plot(Opti.Obj, cols = 3:4, labels = c("Memory Strength", "Density"),
SDT.legend = list(text = c("New", "Old")))
# Plot ROC curve with data
SDT.Plot(Opti.Obj, cols = 3, labels = c("False Alarms", "Hits"), option =
"ROC+")
```

```
# Plot zROC curve with data
SDT.Plot(Opti.Obj, cols = 3, labels = c("z(FA)", "z(H)"), option = "zROC+")
```

This piece of R-code performs the following steps:

1. The data vector with the frequencies of the response catgories for the two stimulus sets (corresponding to the two types of signals) is specified:

```
data <- c(477, 776, 527, 321, 258, 184,   # New
          192, 401, 290, 267, 316, 442)   # Old
```

   For each signal there are 6 response categories. These are orderer from the highest "new" category to the highest "old" category.

2. The estimation function is called:

```
Opti.Obj <- SDT.Estimate(data = data, test = T)
```

   Comment:
   The name of the argument `data` must be specified because for the first argument (`par`) of the function the default value is used. If the argument name were not specified the function would interpret the input as the parameter vector.

   In addition, the test flag is set to `TRUE`. As a result, the gradient is computed numerically at the optimum by means of the function `grad()` in the package `numDeriv` and compared to the analytically computed gradient.
   Comment:
   It is good practice to set `test = TRUE` because non-negligible differences between symbolic and numeric gradients indicate problems of the estimation process, for example, due to the fact that some of the estimates are on the border of the parameter space (e.g. a standard deviation parameter near zero).

3. The result of the estimation is evaluated:

```
Stat.Obj <- SDT.Statistics(Opti.Obj)
```

4. The result is displayed:

```
print(Stat.Obj)
```

5. The configuration is plotted:

```
SDT.Plot(Opti.Obj, cols = 3:4, labels = c("Memory Strength", "Density"),
SDT.legend = list(text = c("New", "Old")))

# Plot ROC curve with data
SDT.Plot(Opti.Obj, cols = 3, labels = c("False Alarms", "Hits"), option =
"ROC+")

# Plot zROC curve with data
SDT.Plot(Opti.Obj, cols = 3, labels = c("z(FA)", "z(H)"), option = "zROC+")


SDT.Plot(Opti.Obj, cols = 3:4, labels = c("Memory Strength", "Density"),
SDT.legend = list(text = c("New", "Old")))
```

   Three plots are created (cf. Figure 12):
   (i) A plot of the density curves;
   (ii) A plot of the ROC curve with the data;
   (iii) A plot of the z-ROC curve with the data.

   Comment:
   A detailed description of the function `SDT.Plot()` may be found in Chapter 2.5.

This piece of R-code results in the following output:

```
=========================================================
 Symbolic and numeric gradients at the optimum:
 Length of difference vector  : 7.55028e-08
 Length of gradient at optimum: 2.441501e-09
=========================================================
          Symbolic Numeric Difference
Mean[2]          0   0e+00      0e+00
Stddev[2]        0   0e+00      0e+00
t-1              0  -4e-08      4e-08
t-2              0   0e+00      0e+00
t-3              0  -5e-08      5e-08
t-4              0   1e-08     -1e-08
t-5              0   3e-08     -3e-08
=========================================================
$Model.description
[1] "Standard SDT model with noise distribution N(0, 1), Number of models:
<2>, Type of restrictions: <no>"

$Statistics
                                        Statistic
log L                                   -7642.024
X^2                                         5.723
G^2                                         5.750
df                                          3.000
p(Y > X^2)                                  0.126
p(Y > G^2)                                  0.124
AIC                                     15298.048
BIC                                     15342.854
CAICF                                   15410.322
ICOMP                                   15288.533
ICOMP.R                                 15288.161
Free Parameters                             7.000
Length of gradient at optimum               0.000
Rank of Hessian                             7.000
Condition number of information matrix      34.932
Rank of model matrix: t(J) * J              7.000
Condition number of model matrix           57.546

$Free.parameters
           Value    SE CFI-95(Lower) CFI-95(Upper)
Mean[2]    0.606 0.036         0.535         0.677
Stddev[2]  1.181 0.034         1.114         1.247
t-1       -0.891 0.028        -0.947        -0.836
t-2       -0.004 0.023        -0.050         0.042
t-3        0.511 0.024         0.464         0.559
t-4        0.928 0.027         0.874         0.981
t-5        1.468 0.035         1.399         1.536

$Full.parametervector
       Gauss-1 Gauss-2
Mean     0.000   0.606
Stddev   1.000   1.181
t-1     -0.891  -0.891
t-2     -0.004  -0.004
t-3      0.511   0.511
t-4      0.928   0.928
t-5      1.468   1.468
```

```
$SDT.measures
       Value    SE CFI-95(Lower) CFI-95(Upper)
d.a[2] 0.554 0.032         0.490         0.619
d.e[2] 0.556 0.033         0.492         0.620
A.z[2] 0.652 0.008         0.636         0.669

$Data.and.Estimates
             Observed Expected Probabilities Residuals  Std.Resid.(analytical)
Signal 1 [1]      477  473.981         0.186     0.139                  0.705
Signal 1 [2]      776  793.236         0.312    -0.612                 -1.340
Signal 1 [3]      527  501.413         0.197     1.143                  2.117
Signal 1 [4]      321  324.712         0.128    -0.206                 -0.332
Signal 1 [5]      258  268.829         0.106    -0.660                 -1.014
Signal 1 [6]      184  180.828         0.071     0.236                  0.633
Signal 2 [1]      192  195.321         0.102    -0.238                 -0.805
Signal 2 [2]      401  382.118         0.200     0.966                  1.606
Signal 2 [3]      290  315.709         0.165    -1.447                 -2.396
Signal 2 [4]      267  265.650         0.139     0.083                  0.138
Signal 2 [5]      316  305.101         0.160     0.624                  1.128
Signal 2 [6]      442  444.101         0.233    -0.100                 -0.554
```

The output is similar to that shown in Chapter 5.1.1. The table at the beginning displays the comparison of the numerically and analytically computed gradients at the optimum. This table is generated by `SDT.Estimate()`.

In the header of the table the difference in length between the numeric and symbolic gradient vector is presented together with the length of the symbolic gradient at the optimum. Ideally, the two length should be close to zero (The length is the square root of the sum of the squared components of the vector). In the present case, both quantities are sufficiently close to zero.
The rows of the table contain the individual components of the symbolic and the numeric gradient vector as well as their difference. Each row corresponds to a free parameter. Again, all entries should be close to zero.

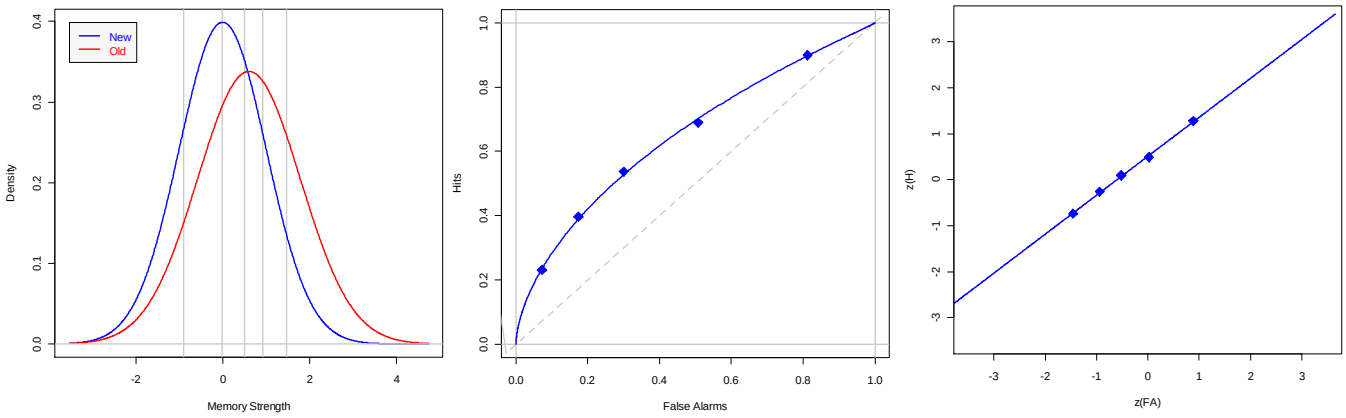
The residual output presents the content of the statistics object `Stat.Obj` that is computed by the procedure `SDT.Statistics()`. The statistics object contains 5 sections:

1. The section `$Model.description` contains a string indicating the model actually used.
2. The section `$Statistics` fit statistics, like Pearson `X^2` and Likelihood ratios statistic `G^2`, as well as other statistics that are relevant for evaluating the quality of the fit, like, for example, the rank and the condition number of the observed information matrix.
3. The section `$Free.parameters` contains the names and values of the estimated parameters and the estimated standard errors (SE) as well as the limits of the confidence intervals.
4. The section `$Full.parametervector` contains the full parameter vector including also those parameters that are subjected to constraints or that are added internally by the program. Each signal distribution makes up a column and the thresholds assigned to the distribution are given below the mean and the standard deviation of the distribution. The standard SDT model comprises only one set of threshold. By consequence, the values of the thresholds for different distributions are identical.
5. The section `$SDT.measures` contains the estimated values, standard errors and confidence intervals for a number of important signal detection measures, like A_z, the area under the estimated ROC curve (for further details, cf. Section 2.2).
6. The section `$Data.and.Estimates` contains the data in the order provided by the input as well as estimated frequencies, probabilities, the Pearson residuals as well as the standardized residuals that are asymptotically standard normal distributed. The row labels `Signal s [r]` indicate the signal (`s`) and the response (`r`) within the signal. For

example, `Signal 1 [6]` indicates the row contains data and estimates for Response Category 6 of Signal 1.

(For further details concerning the contents of the different sections in the output, cf. Chapter 2.2).

The result of the three calls of the plot function `SDT.Plot` is shown in Figure 12.



(a) (b) (c)

Figure 12: *SDT model plots: (a) Density functions and decision bounds of the estimated configuration; (b) ROC plot with data; (c) z-ROC plot with data.*

5.3 Example 3: Fitting data comprising four different types of signal distributions

The present example illustrates the fitting of rating data with four different types of signals using a data set of Ratcliff, McKoon and Tindall (1994, Experiment 1, p. 783). The R-code of this example may be found in the file: `Gauss-4.3 (4 signals, LR intervals).R`
Two different models are fitted:

1. A model with a single set of threshold for the four data sets, and
2. A model with different thresholds for fitting strong and weak items.

The results of the two fits are then compared, using the function `SDT.Anova()`. This type of estimation and model comparison requires the Gaussian version of the SDT model, i.e. the model with entirely free parameters. Here is an extract of the relevant piece of code:

```
n <- 4                      # NUMBER OF SIGNALS

# DATA RATCLIFF ET AL. (1994, Exp.1, 200 and 50 ms pure lists)
data <- c(477, 776, 527, 321, 258, 184,        # Pure Strong New
          192, 401, 290, 267, 316, 442,        # Pure Strong Old
          235, 649, 719, 442, 254, 156,        # Pure Weak   New
          151, 496, 480, 350, 221, 142)        # Pure Weak   Old

# ESTIMATE MODEL
Opti.Obj <- SDT.Estimate(data = data, Model.Id = "Gaussian", n = n, test = T)

# EVALUTE THE RESULTS
Stat.Obj <- SDT.Statistics(Opti.Obj)

# PRINT THE RESULTS
cat("\n=========================================\n")
cat(" Result of the 4 group Gaussian model:")
cat("\n=========================================\n")
print(Stat.Obj)

# NOW ESTIMATE MODELS FOR STRONG AND WEAK ITEMS SEPARATELY
n <- list(n.sdt = 4, restriction = "no")              # New Configuration
```

```
fixed <- matrix(c(0, 1,  0,  1,
                  1, 2, 15, 16), nr = 2, byrow = T)
ident <- matrix(c( 3:7,    (3:7)+14,
                  (3:7)+7, (3:7)+21), nr = 2, byrow = T)

Opti.Obj <- SDT.Estimate(data = data, Model.Id = "Gaussian", n = n, fixed = fixed,
ident = ident, test = T)
Stat2.Obj <- SDT.Statistics(Opti.Obj, control = list(LR = T))

cat("\n=====================================================\n")
cat(" Result of the 4 group Gaussian model:")
cat("\n Strong and weak Items are estimated separately")
cat("\n=====================================================\n")
print(Stat2.Obj)

# COMPARE THE RESTRICTED WITH THE UNRESTRECTED MODEL
cat("\n=============================================================\n")
cat(" Comparison of models with simultaneous and separate fit:")
cat("\n=============================================================\n")
print(SDT.Anova(Stat2.Obj, Stat.Obj))
```

The code performs the following steps:

1. The number of signals is 4:

```
n <- 4                          # NUMBER OF SIGNALS
```

   In the present case this number is passed to the estimation function `SDT.Estimate()` as the configuration information. The number is internally expanded to a full list of configuration information that looks like this:

```
list(n.sdt = 4, restriction = "standard")
```

2. The data vector with the response frequencies is specified:

```
data <- c(477, 776, 527, 321, 258, 184,       # Pure Strong New
          192, 401, 290, 267, 316, 442,       # Pure Strong Old
          235, 649, 719, 442, 254, 156,       # Pure Weak   New
          151, 496, 480, 350, 221, 142)       # Pure Weak   Old
```

3. The estimation procedure `SDT.Estimate()` is called with the model identification string `"Gaussian"`. In addition, the number of signals is passed as configuration information to the function. Finally, the test flag `test` is set to `TRUE` enabling a comparison of numeric and analytic gradients at the optimum.

```
Opti.Obj <- SDT.Estimate(data = data, Model.Id =, n = n, test = T)
```

4. The result is evaluated using the procedure `SDT.Statistics()` and the result is printed:

```
Stat.Obj <- SDT.Statistics(Opti.Obj, control = list(LR = T))
print(Stat.Obj)
```

   The argument `control = list(LR = T)` indicates that likelihood ratio confidence intervals are computed.

5. A second estimation run is performed with strong and weak items being estimated separately. This requires a new specification of the configuration information as well as of fixed and equality constraints. The new configuration information consist in the following list:

```
n <- list(n.sdt = 4, restriction = "no")
```

   In the present case the complete list has to be specified since non of the default options is used.

6. Fixed and equality constraints are specified:

```
fixed <- matrix(c(0, 1,  0,  1,
```

```
                  1, 2, 15, 16), nr = 2, byrow = T)
ident <- matrix(c( 3:7,    (3:7)+14,
                  (3:7)+7, (3:7)+21), nr = 2, byrow = T)
```

In order to check whether these restriction are correct it is useful to use the function that displays the parameter information (For the sake of brevity this was not included into the above code).

```
PI <- SDT.Parameter.Info(data = data, n = n, Model.Id = "Gaussian",
fixed = fixed, ident = ident)
print(PI)
```

This results in the following output:

```
$Model
[1] "Gaussian SDT model with freely estimable parameters for each model,
Number of models: <4>, Type of restrictions: <no>"

$Parameters.and.Constraints
        name   par fixed.value ident.source Nr
1    Mean[1]  0.00           0          ---
2  Stddev[1]  1.00           1          ---
3     t-1[1] -1.50         ---          ---
4     t-2[1] -0.75         ---          ---
5     t-3[1]  0.00         ---          ---
6     t-4[1]  0.75         ---          ---
7     t-5[1]  1.50         ---          ---
8    Mean[2]  0.00         ---          ---
9  Stddev[2]  1.00         ---          ---
10    t-1[2] -1.50         ---       t-1[1]  3
11    t-2[2] -0.75         ---       t-2[1]  4
12    t-3[2]  0.00         ---       t-3[1]  5
13    t-4[2]  0.75         ---       t-4[1]  6
14    t-5[2]  1.50         ---       t-5[1]  7
15   Mean[3]  0.00           0          ---
16 Stddev[3]  1.00           1          ---
17    t-1[3] -1.50         ---          ---
18    t-2[3] -0.75         ---          ---
19    t-3[3]  0.00         ---          ---
20    t-4[3]  0.75         ---          ---
21    t-5[3]  1.50         ---          ---
22   Mean[4]  0.00         ---          ---
23 Stddev[4]  1.00         ---          ---
24    t-1[4] -1.50         ---       t-1[3] 17
25    t-2[4] -0.75         ---       t-2[3] 18
26    t-3[4]  0.00         ---       t-3[3] 19
27    t-4[4]  0.75         ---       t-4[3] 20
28    t-5[4]  1.50         ---       t-5[3] 21
```

The output reveals the following restrictions:

1. The mean and the standard deviations of the first and the third Gaussian distribution are fixed to 0 and 1 (see Parameters 1-2 and 15-16). This corresponds to the fixed constraints given above where the first row of the matrix contains the values and the second row contains the positions of the parameters.
2. The thresholds assigned to the second Gaussian model (Parameters 10-14) are constrained to be equal to the threshold parameters of the first model (Parameters 3-7). Similarly, the thresholds assigned to the fourth Gaussian model (Parameters 24-28) are constrained to be equal to the threshold parameters of the third model (Parameters 17-21). This information is provided in the last column. This corresponds exactly to

the rows of the matrix of identity constraints with the first row containing the positions of the source parameters (3-7 and 17-21) and the second row containing the positions of the target parameters (10-14 and 24-28).

Comments:

(i) Due to the fact that strong and weak items are estimated using different thresholds the location and scale parameters of the third distribution must be fixed (or an equivalent type of constraint has to be imposed). Otherwise the model is not identified and a warning message is printed (cf. Chapter 5.6).

(ii) Detailed information on how to specify constraints on parameters are found in Chapter 3.13. An example concerning the specification of non-linear constraints on parameters is given in Chapter 5.4.

7. The model is estimated a second time with the new configuration and and the results are evaluated and printed.

```
Opti.Obj <- SDT.Estimate(data = data, Model.Id = "Gaussian", n = n, fixed = fixed,
ident = ident, test = T)
Stat2.Obj <- SDT.Statistics(Opti.Obj, control = list(LR = T))
print(Stat2.Obj)
```

8. The results of the two fits are compared using the function `SDT.Anova()`:

```
SDT.Anova(Stat2.Obj, Stat.Obj)
```

The result of the function looks like this:

```
                G2 df     p
Model 1     38.286  9 0.000
Model 2     11.256  6 0.081
Difference 27.030  3 0.000
```

The function provides the likelihood ratio statistic `G2`, the degrees of freedom `df` associated with the statistic as well as the associated `p` value, for the two models as well as the difference between the two models.

In the present case, the difference statistic $\Delta G^2(df = 3) = 27.03$ printed in the last row is not distributed according to a chi-square distribution since the two models are not nested. Thus, the computed *p* value is not valid.

5.4 Example 4: Specifying nonlinear constraints on parameters

The present example demonstrates the specification of nonlinear constraints on parameters by means of a user defined function using the data of Ratcliff, McKoon and Tindall (1994, Experiment 1, p. 783) with four signals. The R-code is similar to that of the previous example. The man difference consists in the specification of the additional non-linear constraint that $\sigma_2 = 1.2 \cdot \sigma_4$, i.e., the standard deviation of the second distribution is 1.2 times that of Distribution 4. The R-code of this example is contained in the file:

```
SDT-3.1 (Functional constraints).R
```

The following extract of the R code illustrates the four components that are required for a proper specification of non-linear constraints on parameters:

1. The function that implements the nonlinear constraints;
2. The assigned gradient function;
3. The function that generates the Jacobian matrix for performing the proper adjustment of the model matrix;
4. The addition of the parameters that are subjected to nonlinear constraints to the matrix of fixed parameters.

```
# FUNCTION PERFOMING FUNCTIONAL CONSTRAINT: s2 = 1.2*s4
fct.s <- function(par)
{
      tindex <- 2                   # Target index: s2
      sindex <- 6                   # Source index: s4

      par[tindex] <- 1.2 * par[sindex]

      # Define Gradient function
      gr.fct <- function(grad, par)
      {
            grad[sindex] <- grad[sindex] + 1.2 * grad[tindex]
            return(grad)
      }
      attr(par, "SDT.gradient") <- gr.fct             # Gradient
      return(par)
}

# FUNCTION COMPUTING THE JACOBIAN MATRIX CORRESPONDING TO FUNCTIONAL
CONSTRAINTS
J.fct <- function(full.par)
{
      J <- diag(length(full.par))
      tindex <- 2
      sindex <- 6
      J[tindex, sindex] <- 1.2
      J
}
# PASS FUNCTION FOR COMPUTING THE JACOBIAN MATRIX AS AN ATTRIBUTE
"Jacobian" TO THE FUNCTION
attr(fct.s, "SDT.Jacobian") <- J.fct

# ADD SPECIFIED PARAMETER TO FIXED ONES
fixed <- matrix(c(1, 2), nrow = 2, byrow = T)

# ESTIMATE MODEL
Opti.Obj <- SDT.Estimate(data = data, n = n, fixed = fixed, functional =
fct.s, test = T)
Stat.Obj <- SDT.Statistics(Opti.Obj)
```

1. The function `fct.s` implements the non-linear functional constraint. This function simply sets the parameter on the 2. position of the parameter vector (= the position of σ_2) equal to 1.2 times the value of the parameter on the 6. position (= the position of σ_4).

2. The gradient function `gr.fct()` is be specified. This function adds the gradient of the log-likelihood function with respect to σ_2, multiplied by 1.2, to the partial derivate of the log-likelihood function with respect to σ_4. The gradient function is passed as an attribute `"SDT.gradient"` to the parameter vector. The gradient function that computes the symbolic gradient recovers it from the attribute of the parameter and uses it for the adjustment of the gradient.
   If no gradient function is specified the estimation procedure outputs an error message and stops. However, models with non-linear constraints on parameters can be estimated without the specification of a gradient function by setting the flag `sym.gr = F` as argument of the estimation function. In this case, gradients and model matrices are computed numerically which slows down the process of estimation.

3. The function `J.fct` is specified. It computes the full Jacobian matrix of the functional constraints. This function is used for the analytic computation of the standardized residuals. It is attached to the function `fct.s` as the attribute `"SDT.Jacobian"`.
   If no function for computing the Jacobian matrix of functional constraints is specified, the model matrix is computed numerically.
4. Parameters subjected to functional constraints must be specified as fixed parameters, i.e., they have to be added to the matrix of fixed constraints:

```
fixed <- matrix(c(1, 2), nrow = 2, byrow = T)
```

5. Finally, the function is passed to the estimation procedure as an additional parameter (alos the matrix of fixed constraints is passed to the function):

```
Opti.Obj <- SDT.Estimate(data = data, n = n, fixed = fixed, functional =
fct.s, test = T)
```

Comment:
A detailed description of how to specify functional constraints is presented in Chapter 4.4.

5.5 Example 5: Fitting data with different number of response categories per signal

The general SDT model with all parameters being freely estimable allows one to fit the model to signals with different number of data points per signal. In this case, a vector with the different number of data for the different signals is passed to the estimation function for parameter `n` (instead of the number of signals). The following R code is contained in the example file: `Gauss-5.2 (Unequal number of data per signal, LR intervals).R` It demonstrates this facility of the model:

```
n.data <- c(7, 4, 3, 6, 2)          # NUMBER OF DATA POINTS PER SIGNAL

# ARTIFICIAL DATA
data <- c( 24,  22,  57,  10,  12,  17,  7,     # Signal 1: 7 data points
          210,  34,   5,   1,                   # Signal 2: 4 data points
           60, 103,   8,                        # Signal 3: 3 data points
            0,   5, 198, 225, 148,  13,         # Signal 4: 6 data points
          121,  69)                             # Signal 5: 2 data points

# FIXED: DISTRIBUTION OF FIRST AND FIFTH SIGNAL = N(0,1)
fixed <- matrix(c(0, 1,  0,  1,
                  1, 2, 25, 26), nr = 2, byrow = T)

# EQUALITY CONSTRAINTS ON THRESHOLDS FOR DIFFERENT SIGNALS:
# 1. CONSTRAINTS ON THRESHOLDS ASSOCIATED WITH SIGNAL 2
ident.2 <- matrix(c(4:6,
                    3:5+8), nr = 2, byrow = T)

# 2. CONSTRAINTS ON THRESHOLDS ASSOCIATED WITH SIGNAL 3
ident.3 <- matrix(c(4:5,
                    3:4+13), nr = 2, byrow = T)

# 3. CONSTRAINTS ON THRESHOLDS ASSOCIATED WITH SIGNAL 4
ident.4 <- matrix(c(3:7,
                    3:7+17), nr = 2, byrow = T)

# 4. CONSTRAINTS ON THRESHOLDS ASSOCIATED WITH SIGNAL 5
ident.5 <- matrix(c(5,
                    3+24), nr = 2, byrow = T)

ident <- cbind(ident.2, ident.3, ident.4, ident.5)
```

```
# ESTIMATE THE MODEL
Opti.Obj <- SDT.Estimate(data = data, Model.Id = "Gaussian", n = n.data,
fixed = fixed, ident = ident, test = T)
Stat.Obj <- SDT.Statistics(Opti.Obj, control = list(LR = T))
```

This piece of code performs the following steps:

1. Specification of the vector with number of data points per signal:

```
n.data <- c(7, 4, 3, 6, 2)          # NUMBER OF DATA POINTS PER SIGNAL
```

   In the present case, there are 5 different types of signals with the first comprising 7 response categories, the second one 4, etc.

2. The data are specified:

```
# ARTIFICIAL DATA
data <- c( 24,  22,  57,  10,  12,  17,  7,     # Signal 1: 7 data points
          210,  34,   5,   1,                   # Signal 2: 4 data points
           60, 103,   8,                        # Signal 3: 3 data points
            0,   5, 198, 225, 148,  13,         # Signal 4: 6 data points
          121,  69)                             # Signal 5: 2 data points
```

3. Fixed constraints are imposed to make the model identifiable:

```
# FIXED: DISTRIBUTION OF FIRST AND FIFTH SIGNAL = N(0,1)
fixed <- matrix(c(0, 1,  0,  1,
                  1, 2, 25, 26), nr = 2, byrow = T)
```

   The first and last signal distribution are specified to be the standard normal distribution $N(0, 1)$.

   Comments:

   - In case of different data points with different signals all the restrictions have to be specified by hand, i.e., the option `restriction` in the configuration list cannot be used. It is always set to `restriction = "no"` by the estimation function.
   - The variance parameter of the last signal distribution has to be fixed in order to make the model identified. This is required, because only one free data point is associated with this type of signal. This precludes the estimation of two free parameters for this distribution.

4. Equality constraints on thresholds are specified: The specification is performed separately for the signal distributions 2-5. These constraints are then collected in one matrix of equality constraints:

```
# EQUALITY CONSTRAINTS ON THRESHOLDS FOR DIFFERENT SIGNALS:
# 1. CONSTRAINTS ON THRESHOLDS ASSOCIATED WITH SIGNAL 2
ident.2 <- matrix(c(4:6,
                    3:5+8), nr = 2, byrow = T)

# 2. CONSTRAINTS ON THRESHOLDS ASSOCIATED WITH SIGNAL 3
ident.3 <- matrix(c(4:5,
                    3:4+13), nr = 2, byrow = T)

# 3. CONSTRAINTS ON THRESHOLDS ASSOCIATED WITH SIGNAL 4
ident.4 <- matrix(c(3:7,
                    3:7+17), nr = 2, byrow = T)

# 4. CONSTRAINTS ON THRESHOLDS ASSOCIATED WITH SIGNAL 5
ident.5 <- matrix(c(5,
                    3+24), nr = 2, byrow = T)

ident <- cbind(ident.2, ident.3, ident.4, ident.5)
```

Comment:
To aid the specification of constraints and the checking of the correctness of the specified constraints the function `SDT.Parameter.Info()` might be helpful (cf. Chapter 2.4 and the example in Chapter 5.3).

5. The model is estimated and the statistics are computed. Note, once again, that the vector `n.data` with the number of response categories per signal is passed to the function for the argument `n`.

```
Opti.Obj <- SDT.Estimate(data = data, Model.Id = "Gaussian", n = n.data,
fixed = fixed, ident = ident, test = T)
Stat.Obj <- SDT.Statistics(Opti.Obj, control = list(LR = T))
```

The following output is produced:

```
=========================================================
 Symbolic and numeric gradients at the optimum:
 Length of difference vector  : 3.839714e-08
 Length of gradient at optimum: 3.820168e-05
=========================================================
            Symbolic    Numeric Difference
t-1[1]    -7.000e-08 -7.000e-08      0e+00
t-2[1]     1.410e-06  1.420e-06      0e+00
t-3[1]     2.741e-05  2.739e-05      2e-08
t-4[1]    -9.520e-06 -9.520e-06      0e+00
t-5[1]     2.200e-07  2.100e-07      0e+00
t-6[1]    -3.000e-08 -4.000e-08      0e+00
Mean[2]   -2.400e-07 -2.400e-07      0e+00
Stddev[2] -4.100e-07 -4.000e-07      0e+00
Mean[3]   -1.200e-06 -1.200e-06      0e+00
Stddev[3] -1.320e-06 -1.340e-06      2e-08
Mean[4]   -1.749e-05 -1.748e-05     -1e-08
Stddev[4]  1.750e-05  1.747e-05      3e-08
=========================================================
============================================================
 Result of the Gaussian model with unequal data per signal:
 Data points per signal: n =( 7 4 3 6 2 )
============================================================
$Model.description
[1] "Gaussian SDT model with freely estimable parameters for each model, data
points per signal: <7, 4, 3, 6, 2>, Type of restrictions: <no>"

$Statistics
                                      Statistic
log L                                -1367.033
X^2                                     13.859
G^2                                     14.041
df                                       5.000
p(Y > X^2)                               0.017
p(Y > G^2)                               0.015
AIC                                   2758.066
BIC                                   2820.551
CAICF                                 2917.785
ICOMP                                 2754.292
ICOMP.R                               2749.214
Free Parameters                         12.000
Length of gradient at optimum            0.000
Rank of Hessian                         12.000
Condition number of information matrix  617.689
Rank of model matrix: t(J) * J          12.000
Condition number of model matrix       1627.429

$Free.parameters
           Value    SE CFI-95(Lower) CFI-95(Upper) LR-95(Lower) LR-95(Upper)
t-1[1]    -0.935 0.116        -1.162        -0.707
```

```
t-2[1]    -0.421 0.089        -0.597        -0.246
t-3[1]     0.332 0.066         0.204         0.460
t-4[1]     0.667 0.077         0.517         0.817
t-5[1]     1.122 0.105         0.915         1.329
t-6[1]     1.772 0.166         1.446         2.098
Mean[2]   -1.143 0.204        -1.543        -0.744       -1.612       -0.791
Stddev[2]  0.727 0.131         0.472         0.983        0.515        1.044
Mean[3]   -0.281 0.087        -0.452        -0.111       -0.456       -0.114
Stddev[3]  0.366 0.048         0.272         0.460        0.282        0.472
Mean[4]    0.460 0.069         0.325         0.595        0.325        0.596
Stddev[4]  0.341 0.035         0.273         0.409        0.277        0.413

$Full.parametervector
       Gauss-1 Gauss-2 Gauss-3 Gauss-4 Gauss-5
Mean     0.000  -1.143  -0.281   0.460   0.000
Stddev   1.000   0.727   0.366   0.341   1.000
t-1     -0.935  -0.421  -0.421  -0.935   0.332
t-2     -0.421   0.332   0.332  -0.421      NA
t-3      0.332   0.667      NA   0.332      NA
t-4      0.667      NA      NA   0.667      NA
t-5      1.122      NA      NA   1.122      NA
t-6      1.772      NA      NA      NA      NA

$Data.and.Estimates
             Observed Expected Probabilities Residuals  Std.Resid.(analytical)
Signal 1 [1]       24   26.076         0.175    -0.406                  -1.516
Signal 1 [2]       22   24.094         0.162    -0.427                  -1.619
Signal 1 [3]       57   43.707         0.293     2.011                   4.151
Signal 1 [4]       10   17.500         0.117    -1.793                  -2.441
Signal 1 [5]       12   18.120         0.122    -1.438                  -1.944
Signal 1 [6]       17   13.815         0.093     0.857                   2.212
Signal 1 [7]        7    5.688         0.038     0.550                   2.212
Signal 2 [1]      210  209.874         0.839     0.009                   0.854
Signal 2 [2]       34   34.807         0.139    -0.137                  -0.854
Signal 2 [3]        5    3.713         0.015     0.668                   0.854
Signal 2 [4]        1    1.605         0.006    -0.478                  -0.854
Signal 3 [1]       60   60.000         0.351     0.000                   0.000
Signal 3 [2]      103  103.000         0.602     0.000                   0.000
Signal 3 [3]        8    8.000         0.047     0.000                   0.000
Signal 4 [1]        0    0.013         0.000    -0.112                  -0.113
Signal 4 [2]        5    2.849         0.005     1.274                   2.059
Signal 4 [3]      198  205.489         0.349    -0.522                  -1.857
Signal 4 [4]      225  220.400         0.374     0.310                   0.914
Signal 4 [5]      148  144.933         0.246     0.255                   0.732
Signal 4 [6]       13   15.317         0.026    -0.592                  -1.501
Signal 5 [1]      121  119.709         0.630     0.118                   0.264
Signal 5 [2]       69   70.291         0.370    -0.154                  -0.264
```

The output reveals two new features:

1. In the section `$Full.parametervector`, the symbols `NA` (= not available) are shown indicating that some decision bounds are not available for signals with a reduced number of response categories.
2. The row names `Signal s [r]` of the matrix in the section `$Data.and.Estimates` indicate that different data points were provided for different signals. The number `s` indicates the signal and the number `r` indicates the response category for the respective signal.

Comment:

The difference between likelihood ratio and normal based confidence intervals is now more pronounced than in the previous examples (though the difference is not as dramatic in the example of the example file `SDT-4.2 (2 signals, LR intervals, bad data).R`).

5.6 Example 6: Detecting problems of identification

Model parameters are not identified if they cannot be uniquely estimated from the data. The program checks for *local identification* (i.e., identification at the maximum of the likelihood function) by investigating the eigenvalues of the observed information matrix (= Hessian matrix at the optimum). In case of detecting a failure of identification, a warning is printed (if the flag `display.warning = TRUE` of the function `SDT.Statistics()` is set, cf. Chapter 2.2), and the standard deviations of the free parameters are not computed.

A more sensitive test of local identification consists in the evaluation of the matrix $\mathbf{J}^{\mathbf{T}} \cdot \mathbf{J}$ (symbolized by `t(J)*J` in the output), where $\mathbf{J}$ is the model matrix (i.e. the Jacobian matrix of the probabilities with respect to the parameters). If this matrix is not of full rank and the warning flag is set (`display.warning = TRUE`) a warning is printed even if no warning is presented on bases of the evaluation of the Hessian matrix.

To demonstrate this problems, we take up again Example 3 of Chapter 5.3 (i.e., the estimation of two separate SDT models on the basis of the four data sets. However, we relax the restriction on the location parameter of the third signal distribution ($\mu_3 = 0$). As a result, the origin of the scale for the second SDT model is not fixed. The code of this example is contained in the file: `Gauss-6 (4 signals not identified).R`.

The only difference to the code of Example 3 consists in a different matrix of fixed constraints. Specifically, the matrix of fixed constraints in Example 3:

```
fixed <- matrix(c(0, 1,  0,  1,
                  1, 2, 15, 16), nr = 2, byrow = T)
```

is replaced by:

```
fixed <- matrix(c(0, 1,  1,
                  1, 2, 16), nr = 2, byrow = T)
```

Thus, the restriction on the mean of the third distribution (Parameter 15) to be equal to zero has been removed.

This results in the following output:

```
=====================================================
 Symbolic and numeric gradients at the optimum:
 Length of difference vector  : 4.171642e-06
 Length of gradient at optimum: 2.320325e-05
=====================================================
            Symbolic    Numeric Difference
t-1[1]    -3.530e-06 -3.610e-06   9.00e-08
t-2[1]    -4.270e-06 -1.300e-07  -4.15e-06
t-3[1]     1.410e-05  1.403e-05   7.00e-08
t-4[1]    -1.477e-05 -1.468e-05  -9.00e-08
t-5[1]     7.540e-06  7.550e-06  -1.00e-08
Mean[2]    2.800e-07  2.100e-07   7.00e-08
Stddev[2] -3.670e-06 -3.670e-06   0.00e+00
Mean[3]    7.000e-08  4.800e-07  -4.10e-07
t-1[3]    -2.700e-07 -2.500e-07  -2.00e-08
t-2[3]    -2.680e-06 -2.670e-06  -1.00e-08
t-3[3]    -3.200e-07 -3.600e-07   4.00e-08
t-4[3]     2.400e-06  2.390e-06   1.00e-08
t-5[3]    -7.000e-07 -7.300e-07   3.00e-08
Mean[4]    1.500e-06  1.370e-06   1.30e-07
Stddev[4] -2.110e-06 -2.100e-06  -1.00e-08
=====================================================

===> WARNING: Jacobian matrix [t(J) * J] is not positive definite: Rank = 14

$Model.description
[1] "SDT model with arbitrary parameters for each model"
```

```
                                          Statistic
log L                                  -1.479231e+04
X^2                                     1.122600e+01
G^2                                     1.125600e+01
df                                      5.000000e+00
p(Y > X^2)                              4.700000e-02
p(Y > G^2)                              4.700000e-02
AIC                                     2.961462e+04
BIC                                     2.972076e+04
CAICF                                   2.983488e+04
ICOMP                                   2.999515e+04
ICOMP.R                                 2.976975e+04
Free Parameters                         1.500000e+01
Length of gradient at optimum           0.000000e+00
Rank of Hessian                         1.500000e+01
Condition number of information matrix  4.592497e+14
Rank of model matrix: t(J) * J          1.400000e+01
Condition number of model matrix        2.300654e+17

$Free.parameters
           Value        SE CFI-95(Lower) CFI-95(Upper)
t-1[1]    -0.891     0.028        -0.947        -0.836
t-2[1]    -0.004     0.023        -0.050         0.042
t-3[1]     0.511     0.024         0.464         0.559
t-4[1]     0.928     0.027         0.874         0.981
t-5[1]     1.468     0.035         1.399         1.536
Mean[2]    0.606     0.036         0.535         0.677
Stddev[2]  1.181     0.034         1.114         1.247
Mean[3]   -0.184 77800.159   -152485.693    152485.325
t-1[3]    -1.506 77800.160   -152487.018    152484.007
t-2[3]    -0.523 77800.161   -152486.037    152484.992
t-3[3]     0.199 77800.161   -152485.315    152485.713
t-4[3]     0.774 77800.161   -152484.740    152486.288
t-5[3]     1.346 77800.160   -152484.167    152486.858
Mean[4]   -0.106 77800.159   -152485.616    152485.403
Stddev[4]  1.022     0.026         0.971         1.074
[...]
```

Due to fact that the rank of Jacobian matrix multiplied by its transpose $\mathbf{J}^{\mathbf{T}} \cdot \mathbf{J}$ is lower than the number of free parameters and the flag `display.warning` of the `SDT.Statistics` function is set to `TRUE`, a warning is printed indicating that some of the parameters are not identified.

The huge condition number of the Hessian matrix as well as the Jacobian matrix (multiplied by its transpose) as well as the gigantic values of the estimated standard errors (and the resulting confidence limits) indicate the existence of a problem of identification.
This output also indicates that the first two distributions and the associated set of thresholds are not affected by the problem.

Comments:

- The column rank of the matrix $\mathbf{J}^{\mathbf{T}} \cdot \mathbf{J}$ was 14 (one less than the number of free parameters) whereas the rank of the Hessian was 15. This confirms the simulation results of McDonald and Krane (1979) that the Jacobian matrix is more sensitive and thus better suited for detecting problems of identification.
- Note also that the estimation procedure converged properly as indicated by the small value of the gradient at the optimum.

5.7 List of example files for the SDT and Gaussian model

The following list contains all the example files concerning the SDT and Gaussian model. These example files are available from the web site.

```
Gauss-1.1 (Y-N-Data, standard).R
Gauss-1.2 (Y-N-Data, symmetric).R
Gauss-2.1 (2 signals, Data Ratcliff).R
Gauss-2.2 (2 signals, Symmetric configuration).R
Gauss-3 (Functional constraints).R
Gauss-4.1 (2 signals, LR intervals).R
Gauss-4.2 (2 signals, symmetric, LR intervals).R
Gauss-4.3 (4 signals, LR intervals).R
Gauss-5.1 (Gaussian with unequal number of data per signal).R
Gauss-5.2 (Unequal number of data per signal, LR intervals).R
Gauss-6 (4 signals not identified).R
Gauss-7 (ROC Plots).R
SDT-1 (Y-N-Data).R
SDT-2.1 (2 signals, Data Wickens).R
SDT-2.2 (2 signals, Data Ratcliff).R
SDT-2.3 (3 signals, Data Ratcliff).R
SDT-3.1 (Functional constraints).R
SDT-3.2 (Functional constraints, symmetric thresholds).R
SDT-4.1 (2 signals, LR intervals).R
SDT-4.2 (2 signals, LR intervals, bad data).R
SDT-4.3 (4 signals, LR intervals).R
SDT-5 (ROC Plots).R
SDT-Gauss (Parameter Information).R
```

6. Working Examples: Gaussian mixture model (MIX.PD)

The present chapter provides examples for fitting the Gaussian mixture model with a pair of signals represented by three Gaussian distributions (cf. Chapter 3.3). The application of the model is illustrated with two different set of data:

(i) Source monitoring data from Hilford et al. (2002)
(ii) Data on associative recognition from Kelley and Wixted (2001).

6.1 Example 1: Fitting the MIX.PD model to source monitoring data

The file `MIX-PD.1-2 (Source Monitoring, Hilford).R` contains the R-code for fitting the data from the three experiments of Hilford et al. (2002). The code looks like this:

```
# ============================================================
# Examples of the documentation of Model MIX-PD:
# MIX-PD.1-2: Source monitoring with two signals
# Data from Hilford et al. (2002)
# Date of creation: July, 2009
# Author: Siegfried Macho
# ============================================================
library(numDeriv)

# DIRECTORIES (PLEASE ADJUST TO YOUR PERSONAL SETTING)
DirSource <- "C:/Documents and Settings/macho/Statistik/_SDT-Models
(Version 2)/R-Code/"
DirSourceM <- "C:/Documents and Settings/macho/Statistik/_SDT-Models
(Version 2)/R-Code/Models/"

# FILE NAME WITH THE MODEL
Model <- "SDT-MIX-PD.R"

# FILE NAMES FOR ESTIMATION
```

```
Main      <- "SDT-Main.R"
Auxiliary <- "SDT-Auxiliary.R"

# LOAD SOURCE FILES WITH SDT MODEL
source(paste(DirSourceM, Model, sep = ""))

# LOAD SOURCE FILES WITH AUXILIARY AND ESTIMATION FUNCTIONS
source(paste(DirSource, Auxiliary, sep = ""))
source(paste(DirSource, Main, sep = ""))


# DATA: HILFORD ET AL. (2002), Exp.1
data.1 <- c(264, 133, 159, 116,  80,  33,             # Male voice
             25,  62, 105, 128, 145, 294)             # Female voice


# DATA: HILFORD ET AL. (2002), Exp.2
data.2 <- c(1358, 550, 701, 585, 466,  300,           # Male
             278, 477, 598, 614, 589, 1404)           # Female


# DATA: HILFORD ET AL. (2002), Exp.3
data.3 <- c(1313, 538, 524, 529, 363,  333,           # Top
             348, 356, 455, 552, 510, 1379)           # Bottom


# SPECIFY EQUALITY CONSTRAINTS: IDENTICAL PROBABILITIES AND d'
ident <- matrix(c( 1, 3,
                   2, 4), nr = 2, byrow = T)

cat("\n--------------------------------------------------------------------\n")
cat(" MIX-PD: Hilford et al. (2002): Exp.1")
cat("\n--------------------------------------------------------------------\n")
# ESTIAMTE THE MODEL AND EVALUATE THE RESULT
Opti1.Obj <- SDT.Estimate(data = data.1, Model.Id = "MIX.PD", ident =
ident, test = T)
Stat1.Obj <- SDT.Statistics(Opti1.Obj)
print(Stat1.Obj)

cat("\n--------------------------------------------------------------------\n")
cat(" MIX-PD: Hilford et al. (2002): Exp.2")
cat("\n--------------------------------------------------------------------\n")
# ESTIAMTE THE MODEL AND EVALUATE THE RESULT
Opti2.Obj <- SDT.Estimate(data = data.2, Model.Id = "MIX.PD", ident =
ident, test = T)
Stat2.Obj <- SDT.Statistics(Opti2.Obj)
print(Stat2.Obj)

cat("\n--------------------------------------------------------------------\n")
cat(" MIX-PD: Hilford et al. (2002): Exp.3")
cat("\n--------------------------------------------------------------------\n")
# ESTIMATE THE MODEL AND EVALUATE THE RESULT
Opti3.Obj <- SDT.Estimate(data = data.3, Model.Id = "MIX.PD", ident =
ident, test = T)
Stat3.Obj <- SDT.Statistics(Opti3.Obj)
print(Stat3.Obj)

# PLOT GAUSSIAN DISTRIBUTIONS AND THRESHOLDS, AS WELL AS ROC CURVES FOR
EXP.3:
```

```
# PLOT DENSITY CURVES
SDT.Plot(Opti3.Obj, cols = c(3, 4, 6), SDT.legend = list(text = c( "no
info", "left",  "right")))

# PLOT ROC CURVES
# Plot ROC curve with data
SDT.Plot(Opti3.Obj, cols = 3, labels = c("False Alarms", "Hits"), option =
"ROC+")

# Plot z-ROC curve with data
SDT.Plot(Opti3.Obj, cols = 3, labels = c("z(FA)", "z(H)"), option =
"zROC+")
```

The code consists of five sections.

1. The relevant source files (`SDT-Main.R`, `SDT-Auxiliary.R`, and `SDT-MIX-PD.R`) are loaded. This section is contained within the lines:

```
# DIRECTORIES (PLEASE ADJUST TO YOUR PERSONAL SETTING)
...
source(paste(DirSource, Main, sep = ""))
```

2. The three data sets are specified: This section is contained within the lines:

```
# DATA: HILFORD ET AL. (2002), Exp.1
...
              348, 356, 455, 552, 510, 1379)           # Bottom
```

3. Equality (identity) constraints are specified, representing the restrictions: $\pi_{1_{left}} = \pi_{1_{right}}$ and $d'_{1_{left}} = d'_{1_{right}}$:

```
# SPECIFY EQUALITY CONSTRAINTS: IDENTICAL PROBABILITIES AND d'
ident <- matrix(c( 1, 3,
                   2, 4), nr = 2, byrow = T)
```

4. The three data sets are estimated, statistics are computed, and the result is printed. For example, for modeling the data of Experiment 1, the following piece of code is relevant:

```
Opti1.Obj <- SDT.Estimate(data = data.1, Model.Id = "MIX.PD", ident =
ident, test = T)
Stat1.Obj <- SDT.Statistics(Opti1.Obj)
print(Stat1.Obj)
```

   The first command performs the estimation. The results of this process are stored in the object `Opti1.Obj`. Note that the number of signals ($n = 2$) was not passed to the estimation procedure because $n = 2$ is the default value.
   The second command results in the computation of relevant statistics that are stored in the object `Stat1.Obj`.
   Finally, the content of `Stat1.Obj` is printed.

   The commands for estimating the other data sets are analogously. They main difference consists in the fact that the data sets `data.2` and `data.3` are passed (instead of `data.1`) to the estimation procedure.

1. Three plots are provided:
   (i) The configuration of density curves is plotted:

```
SDT.Plot(Opti3.Obj, cols = c(3, 4, 6), SDT.legend = list(text = c( "no
info", "left",  "right")))
```

   (ii) The ROC curves with the data are plotted:

```
SDT.Plot(Opti3.Obj, cols = 3, labels = c("False Alarms", "Hits"), option =
"ROC+")
```

(iii) The *z*-ROC curves with the data are plotted:

```
SDT.Plot(Opti3.Obj, cols = 3, labels = c("z(FA)", "z(H)"), option =
"zROC+")
```

The output for the data of Exp.1 looks like this:

```
------------------------------------------------------------------------------
 MIX-PD: Hilford et al. (2002): Exp.1
------------------------------------------------------------------------------

=========================================================
 Symbolic and numeric gradients at the optimum:
 Length of difference vector  : 1.996004e-07
 Length of gradient at optimum: 1.394832e-05
=========================================================
            Symbolic    Numeric Difference
p.left[1]   1.100e-07  2.200e-07   -1.0e-07
d'.left[1] -7.300e-07 -7.300e-07    0.0e+00
t-1         5.000e-07  5.000e-07    0.0e+00
t-2        -1.200e-06 -1.200e-06   -1.0e-08
t-3         5.910e-06  5.740e-06    1.7e-07
t-4        -1.113e-05 -1.116e-05    3.0e-08
t-5         5.790e-06  5.790e-06    0.0e+00
=========================================================
$Model.description
[1] "Mixture Model: Mixtures of three normal distributions for each pair of
signals"

$Statistics
                                       Statistic
log L                                  -2482.840
X^2                                        2.113
G^2                                        2.108
df                                         3.000
p(Y > X^2)                                 0.549
p(Y > G^2)                                 0.550
AIC                                     4979.680
BIC                                     5017.075
CAICF                                   5071.222
ICOMP                                   4973.280
ICOMP.R                                 4970.388
Free Parameters                            7.000
Rank of Hessian                            7.000
Length of gradient at optimum              0.000
Condition number of information matrix     80.891
Rank of model matrix: t(J) * J             7.000
Condition number of model matrix         165.849

$Free.parameters
            Value SE(delta method) SE(numerical) CFI-95(Lower) CFI-95(Upper)
p.left[1]   0.474            0.029         0.029         0.417         0.532
d'.left[1]  1.927            0.177         0.177         1.581         2.273
t-1        -1.543            0.086         0.086        -1.712        -1.374
t-2        -0.835            0.056         0.056        -0.946        -0.725
t-3        -0.078            0.048         0.048        -0.172         0.015
t-4         0.603            0.053         0.053         0.500         0.707
t-5         1.373            0.080         0.080         1.217         1.530

$Full.parametervector
         Mixture-1
Mean         0.000
Stddev       1.000
p.left       0.474
p.right      0.474
```

```
d'.left      1.927
d'.right     1.927
t-1         -1.543
t-2         -0.835
t-3         -0.078
t-4          0.603
t-5          1.373

$Data.and.Estimates
          Observed Expected Probabilities Residuals  Std.Resid.(analytical)
Data[1, 1]      264  267.246         0.340    -0.199                  -0.662
Data[1, 2]      133  137.245         0.175    -0.362                  -0.930
Data[1, 3]      159  149.318         0.190     0.792                   1.436
Data[1, 4]      116  116.367         0.148    -0.034                  -0.055
Data[1, 5]       80   79.653         0.101     0.039                   0.080
Data[1, 6]       33   35.172         0.045    -0.366                  -0.683
Data[2, 1]       25   24.591         0.032     0.083                   0.149
Data[2, 2]       62   56.926         0.075     0.672                   1.192
Data[2, 3]      105  113.532         0.150    -0.801                  -1.440
Data[2, 4]      128  128.274         0.169    -0.024                  -0.040
Data[2, 5]      145  146.129         0.193    -0.093                  -0.239
Data[2, 6]      294  289.548         0.381     0.262                   0.848
```

The table at the beginning of the output is presented because the flag `test=T` was set in the estimation procedure. By consequence, the numeric gradient at the optimum is computed using the function `grad()` from the R package `numDeriv` that was includes at the beginning of the code file.
The numeric and symbolic gradients are very close, and, in addition, the length of the gradient at the optimum is close to zero, indicating that a stationary point was found.

The model identification string indicates that the correct model was used and the statistics sections provides the relevant general statistical information (for details, cf. Chapter 2.2).

The next section in the output lists the free parameters and their estimated values as well as the estimated standard errors. In the present case, only $\pi_{1_{left}}$ and $d'_{1_{left}}$ as well as the thresholds were estimated since the restrictions $\mu_1 = 0$ and $\sigma_1 = 0$ are set by the program, due to the fact that no foil distribution was present (indicated by the even number of signals). In addition, the parameters $\pi_{1_{right}}$ and $d'_{1_{right}}$ were subjected to equality constraints and were thus not estimated (The values of these parameters are shown in the subsequent section of the output that presents the full parameter vector).
The standard errors of parameters are estimated in two ways:

1. Using the Hessian matrix that is provided by the estimation procedure together with the delta method.
2. Computation of the Hessian, using the procedure `hessian()` from the R package `numDeriv`.

The resulting estimates are shown in the columns named `SE(delta method)` and `SE(numerical)`, respectively. Conflicting values indicate a problem of the estimation. This may happen, for instance, if one of the probability parameters is close to the limit (i.e. near 0.0 or 1.0).
In the present case, the estimates resulting from the two methods are identical. So there seems to be no problem.
The last two columns of the section contain the lower and upper bound of the 95% confidence interval of estimated parameters. These are based on the standard error computed by means of the delta method.

The next section presents the full parameter vector. Inspection reveals that the intended equality constraints were set correctly.

The final section provides the observed data, the estimated model frequencies and probabilities as well as the Pearson and the standardized residuals (cf. Chapter 2.2). The numbers in brackets on the left indicate the number of the signal and the response category, e.g., `Data[2, 4]` refers to the second signal and the forth response category.

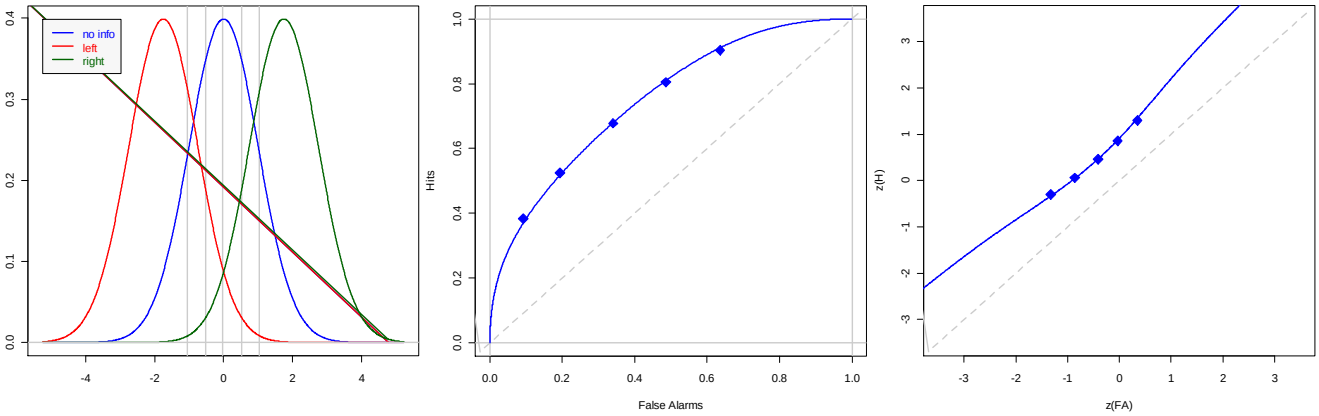
The plots generated by the three calls of the function SDT.Plot is shown in Figure 13.

6.2 Example 2: Fitting the MIX.PD model to associative recognition data

The files:

```
MIX-PD.2-1 (Associative Recognition, Kelly & Wixted, Exp.1).R
MIX-PD.2-2 (Associative Recognition, Kelly & Wixted, Exp.2).R
MIX-PD.2-3 (Associative Recognition, Kelly & Wixted, Exp.3).R
```

contain the R-code for fitting the data from the three experiments of Kelley & Wixted (2001) on associative recognition. The following excerpt from `MIX-PD.2-2 (Associative Recognition, Kelly & Wixted, Exp.2).R` contains the portion of the code for fitting one model to the data of Experiment 2:



(a) (b) (c)

Figure 13: *MIX.PD model: Plot of data and model predictions of Exp.3 of Hilford et al (2003): (a) Configuration of densities and decision bounds; (b) ROC curves, (c) z-ROC curves.*

```
# ================================================================
# Examples of the documentation of Model MIX-PD:
# MIX-PD.2-2: Modeling Associative Recognition
# Data from Kelley and Wixted, Exp.2
# Date of creation: August, 2009
# Author: Siegfried Macho
# ================================================================
... [Code for loading relevant source files]

# DATA OF KELLEY & WIXTED (2001), Experiment 2
data <- c(879, 553, 345,  61,  54,  28,   # Exp.2: Lure items
          179, 109,  82,  45,  28,  37,   # Exp.2: Weak   Items, rearranged
           81,  69,  62,  31,  37, 200,   # Exp.2: Weak   Items, intact
          215,  82,  60,  31,  31,  61,   # Exp.2: Strong Items, rearranged
           21,  24,  33,  21,  33, 348)   # Exp.2: Strong Items, intact

# NUMBER OF CONDITIONS (TYPES OF SIGNALS)
n <- 5
```

```
# IDENTITY CONSTRAINTS. d'.left = d'.right (strong = weak)
ident.d.w.s <- matrix(c(5,   5,   5,
                        6, 5+6, 6+6), nr = 2, byrow = T)

... [Code for fitting other models]

cat("\n-------------------------------------------------------------------\n")
cat("MIX-PD: Associative recognition (Data from Kelley and Wixted, Exp.2):")
cat("\n p.left  <> p.right, d'.left  = d'.right (weak = strong)")
cat("\n-------------------------------------------------------------------\n")
Opti.Obj <- SDT.Estimate(data = data, Model.Id = "MIX.PD", n = n, ident =
ident.d.w.s, test = T)
Stat.Obj <- SDT.Statistics(Opti.Obj)
print(Stat.Obj)

# PLOT GAUSSIAN DISTRIBUTIONS AND THRESHOLDS, AS WELL AS ROC CURVES:
# PLOT DENSITY CURVES
SDT.Plot(Opti.Obj, cols = c(6, 3, 3, 3, 4, 4, 4), SDT.legend = list(text =
c("New", paste("Weak", 1:3),paste("Strong", 1:3)), cols = c(6, 3, 3, 3, 4,
4, 4)))

# PLOT ROC CURVES
# Plot ROC curve with data
SDT.Plot(Opti.Obj, cols = c(3, 3, 4, 4), ltys = c(2, 1, 2, 1), labels =
c("False Alarms", "Hits"), option = "ROC+",
SDT.legend = list(text = c("weak rearranged", "weak intact","strong
rearranged", "strong intact"), cols = c(3, 3, 4, 4), ltys = c(2, 1, 2, 1)))

# Plot z-ROC curve with data
SDT.Plot(Opti.Obj, cols = c(3, 3, 4, 4), ltys= c(2, 1, 2, 1), labels =
c("z(FA)", "z(H)"), option = "zROC+",
SDT.legend = list(text = c("weak rearranged", "weak intact","strong
rearranged", "strong intact"), cols = c(3, 3, 4, 4), ltys = c(2, 1, 2, 1)))
```

Again, the code consists of four main sections.

1. The relevant source files (`SDT-Main.R`, `SDT-Auxiliary.R`, and `SDT-MIX-PD.R`) are loaded. This section was omitted:

```
... [Code for loading relevant source files]
```

2. The five data sets are specified and the number of models are specified:

```
# DATA OF KELLEY & WIXTED (2001), Experiment 2
data <- c(879, 553, 345,  61,  54,  28,   # Exp.2: Lure items
          179, 109,  82,  45,  28,  37,   # Exp.2: Weak   Items, rearranged
           81,  69,  62,  31,  37, 200,   # Exp.2: Weak   Items, intact
          215,  82,  60,  31,  31,  61,   # Exp.2: Strong Items, rearranged
           21,  24,  33,  21,  33, 348)   # Exp.2: Strong Items, intact

# NUMBER OF CONDITIONS (TYPES OF SIGNALS)
n <- 5
```

Note that at the beginning the responses for the lure items is given. The data for lure items have always to be presented before the item pairs.

3. Equality (identity) constraints are specified, representing the restrictions: $d'_{1_{left}} = d'_{1_{right}} = d'_{2_{left}} = d'_{2_{right}}$:

```
# IDENTITY CONSTRAINTS. d'.left = d'.right (strong = weak)
```

```
ident.d.w.s <- matrix(c(5,   5,   5,
                        6, 5+6, 6+6), nr = 2, byrow = T)
```

4. The data are estimated, statistics are computed and the result is printed:

```
Opti.Obj <- SDT.Estimate(data = data, Model.Id = "MIX.PD", n = n, ident =
ident.d.w.s, test = T)
Stat.Obj <- SDT.Statistics(Opti.Obj)
print(Stat.Obj)
```

   The first command performs the estimation. The results of this process are stored in the object `Opti.Obj`. Contrary to the first example with only two data sets, the number of signals must be passed to the estimation procedure (Since the default number of models is 2, this was unnecessary for the previous example).
   The second command results in the computation of relevant statistics that are stored in the object `Stat.Obj`.
   The next command prints the content of `Stat.Obj` is printed.

5. Three plots are provided:
   (i) The configuration of density curves is plotted:

```
SDT.Plot(Opti.Obj, cols = c(6, 3, 3, 3, 4, 4, 4), SDT.legend = list(text =
c("New", paste("Weak", 1:3),paste("Strong", 1:3)), cols = c(6, 3, 3, 3, 4,
4, 4)))
```

   (ii) The ROC curves with the data are plotted:

```
SDT.Plot(Opti.Obj, cols = c(3, 3, 4, 4), ltys = c(2, 1, 2, 1), labels =
c("False Alarms", "Hits"), option = "ROC+",
SDT.legend = list(text = c("weak rearranged", "weak intact","strong
rearranged", "strong intact"), cols = c(3, 3, 4, 4), ltys = c(2, 1, 2, 1)))
```

   (iii) The *z*-ROC curves with the data are plotted:

```
SDT.Plot(Opti.Obj, cols = c(3, 3, 4, 4), ltys= c(2, 1, 2, 1), labels =
c("z(FA)", "z(H)"), option = "zROC+",
SDT.legend = list(text = c("weak rearranged", "weak intact","strong
rearranged", "strong intact"), cols = c(3, 3, 4, 4), ltys = c(2, 1, 2, 1)))
```

The output provided by this piece of code looks like this:

```
------------------------------------------------------------------------
 MIX-PD: Associative recognition (Data from Kelley and Wixted, Exp.2):
 p.left  <> p.right, d'.left  = d'.right (weak = strong)
------------------------------------------------------------------------

=========================================================
 Symbolic and numeric gradients at the optimum:
 Length of difference vector  : 9.843447e-08
 Length of gradient at optimum: 3.152847e-05
=========================================================
             Symbolic    Numeric Difference
Mean[1]     1.415e-05  1.417e-05     -1e-08
Stddev[1]   6.770e-06  6.790e-06     -2e-08
p.left[1]  -8.800e-07 -8.700e-07     -1e-08
p.right[1]  4.190e-06  4.230e-06     -5e-08
d'.left[1]  3.960e-06  3.950e-06      1e-08
Mean[2]     6.870e-06  6.900e-06     -4e-08
Stddev[2]  -2.400e-07 -2.600e-07      1e-08
p.left[2]  -1.350e-06 -1.320e-06     -2e-08
p.right[2]  1.650e-06  1.650e-06      0e+00
t-1         7.050e-06  7.080e-06     -3e-08
t-2        -2.276e-05 -2.271e-05     -5e-08
t-3        -5.870e-06 -5.900e-06      3e-08
t-4        -6.010e-06 -6.020e-06      1e-08
t-5        -4.910e-06 -4.920e-06      1e-08
```

```
=======================================================
$Model.description
[1] "Mixture Model: Mixtures of three normal distributions for each pair of
signals"

$Statistics
                                        Statistic
log L                                   -5249.008
X^2                                        13.004
G^2                                        12.867
df                                         11.000
p(Y > X^2)                                  0.293
p(Y > G^2)                                  0.302
AIC                                     10526.017
BIC                                     10613.562
CAICF                                   10709.538
ICOMP                                   10545.679
ICOMP.R                                 10515.094
Free Parameters                            14.000
Rank of Hessian                            14.000
Length of gradient at optimum               0.000
Condition number of information matrix   12191.540
Rank of model matrix: t(J) * J             14.000
Condition number of model matrix         6849.359

$Free.parameters
            Value SE(delta method) SE(numerical) CFI-95(Lower) CFI-95(Upper)
Mean[1]     0.509            0.123         0.123         0.268         0.749
Stddev[1]   1.238            0.078         0.078         1.084         1.391
p.left[1]   0.099            0.066         0.066         0.000         0.229
p.right[1]  0.461            0.058         0.058         0.348         0.575
d'.left[1]  2.720            0.307         0.307         2.118         3.322
Mean[2]     0.850            0.543         0.543        -0.215         1.915
Stddev[2]   1.475            0.308         0.309         0.871         2.080
p.left[2]   0.306            0.214         0.214         0.000         0.726
p.right[2]  0.843            0.141         0.141         0.566         1.000
t-1        -0.108            0.029         0.029        -0.164        -0.052
t-2         0.675            0.029         0.029         0.618         0.733
t-3         1.402            0.039         0.039         1.326         1.478
t-4         1.746            0.047         0.047         1.654         1.838
t-5         2.187            0.063         0.063         2.065         2.310

$Full.parametervector
           Foil Mixture-1 Mixture-2
Mean      0.000     0.509     0.850
Stddev    1.000     1.238     1.475
p.left       NA     0.099     0.306
p.right      NA     0.461     0.843
d'.left      NA     2.720     2.720
d'.right     NA     2.720     2.720
t-1      -0.108    -0.108    -0.108
t-2       0.675     0.675     0.675
t-3       1.402     1.402     1.402
t-4       1.746     1.746     1.746
t-5       2.187     2.187     2.187

$Data.and.Estimates
...
```

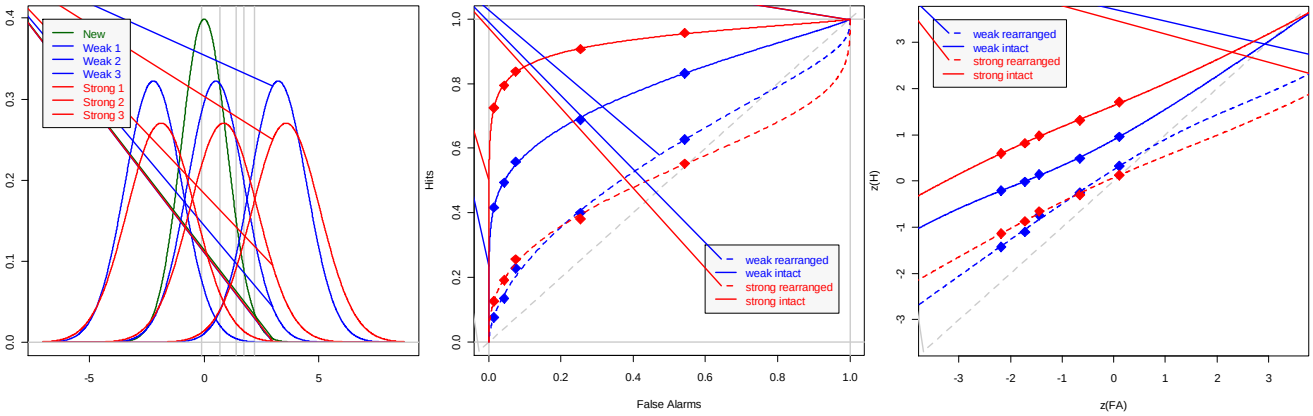
The table at the beginning of the output is presented because the flag `test=T` was set in the estimation procedure. By consequence, the numeric gradient at the optimum is computed using the function `grad()` from the R package `numDeriv` that was includes at the beginning of the code file.

As in the previous example, the numeric and symbolic gradients are very close, and, in addition, the length of the gradient at the optimum is close to zero, indicating that a stationary point was found.

The model identification string indicates that the correct model was used and the statistics sections provides the relevant general statistical information (for details, cf. Chapter 2.2). The value of G^2 statistic corresponds exactly to that reported in Macho (2004, Table 3 on p.89). The latter estimations were performed by using the solver of the program Excel. Both programs lead to the same result (this was true for all fits performed).

The next section lists the free parameters, estimated values, estimated standard errors, and confidence limits of the 95% confidence interval. In the present case, due to the equality restrictions specified, only $d'_{1_{left}}$ was estimated, but not $d'_{1_{right}}$, $d'_{2_{left}}$ and $d'_{2_{right}}$. In addition, due to the presence of a foil distribution, μ_1 and σ_1 (the mean and standard deviation of the Gaussian corresponding to the first pair of stimuli) have been estimated.
The standard errors of parameters are estimated by the two methods described above (cf. Chapter 6.1). Both estimates of standard errors are identical indicating the absence of major problems. Note, however, that some of the confidence intervals of the probability parameters include, respectively, the minimal (0.0) and maximal (1.0) possible value.

The next section presents the full parameter vector. The three columns labeled `Foil Mixture-1 Mixture-2` contain the estimates for the foil distribution and for the two mixture models representing strong and weak pairs. The `NA` for foil distribution indicate that the respective parameters are not present. The last rows comprise the estimated thresholds. By assumption these are identical for each of the partial models.



(a) (b) (c)

Figure 14: *MIX.PD model: Plot of data and model predictions of Kelley and Wixted (2001, Exp.2): (a) Configuration of densities and decision bounds; (b) ROC curves, (c) z-ROC curves.*

Comment:

Note that the condition number of the information matrix as well as the Jacobian matrix are rather high. In addition, some of the estimated standard errors are also relatively high. This indicates that it would be favorable to specify further constraints, e.g. equality constraints on the means and standard deviations of the mixtures for strong and weak pairs.

Figure 14 exhibits the plots generated by the three calls of the plot command.

7. Working Examples: Gaussian mixture model with two distributions per signal (MIX.2)

The present chapter provides examples for fitting the Gaussian mixture model with two distributions available per signal (cf. Chapter 3.4). This model though extremely powerful usually requires the specification of complex constraints. Thus, for simple applications it is favorable to employ the MIX.PD model which is much easier to use. The application of the model is illustrated by means of three examples:

(1) The mixture model of DeCarlo (2007) is fitted to the data on the mirror effect of Exp.1B of DeCarlo (2007);
(2) The source monitoring model is fitted to the source monitoring data from Hilford et al. (2002);
(3) The data from Experiment 2 of Kelley and Wixted (2001) are fitted.

7.1 Example 1: Fitting the MIX.2 model to data on the mirror effect

The file `MIX-2.5.1 (MIX-2 Mirror effect, DeCarlo, 2007).R` contains the R-code for fitting the data from Experiment 1A and 1B of DeCarlo (2007) on the mirror effect. The model for modeling these data does not require the specification of complex functional constraints (The same is true for the mixture models of DeCarlo (2008) for modeling process dissociation data (cf. the file: `MIX-2.4 (MIX-2 Process Dissociation, Yonelinas, 1994).R`)).

This is the R-Code:

```
# ============================================================
# MIX-2.5.1: Examples of the documentation of Model MIX.2:
# Modeling the mirror effect
#
# Model: DeCarlo (2007), Model 2
# Data: DeCarlo (2007)
#
# Date of creation: May, 2010
# Author: Siegfried Macho
# ============================================================
library(numDeriv)

# DIRECTORIES (PLEASE ADJUST TO YOUR PERSONAL SETTING)
DirSource <- "C:/Documents and Settings/macho/Statistik/_SDT-Models
(Version 2)/R-Code/"
DirSourceM <- "C:/Documents and Settings/macho/Statistik/_SDT-Models
(Version 2)/R-Code/Models/"

# FILE NAME WITH THE MODEL
Model <- "SDT-MIX-2.R"

# FILE NAMES FOR ESTIMATION
Main      <- "SDT-Main.R"
Auxiliary <- "SDT-Auxiliary.R"

# LOAD SOURCE FILES WITH SDT MODEL
source(paste(DirSourceM, Model, sep = ""))

# LOAD SOURCE FILES WITH AUXILIARY AND ESTIMATION FUNCTIONS
source(paste(DirSource, Auxiliary, sep = ""))
source(paste(DirSource, Main, sep = ""))

# NUMBER OF SIGNALS
n <- 4
```

```
# DATA: DeCARLO (2007) Exp.1A: NATIVE SPEAKERS
data.1A <- c( 721, 1418, 1460,  860, 422,  159,        # High frequency New
             1457, 1436,  991,  567, 398,  191,        # Low frequency New
              211,  689, 1016, 1028, 894, 1202,        # High frequency Old
              233,  457,  568,  729, 995, 2058)        # Low frequency Old


# DATA: DeCARLO (2007) Exp.1B: NON-NATIVE SPEAKERS
data.1B <- c(740, 791, 616, 390, 230,  173,        # High frequency New
             899, 755, 504, 344, 236,  202,        # Low frequency New
             317, 449, 448, 374, 405,  947,        # High frequency Old
             215, 305, 309, 338, 478, 1295)        # Low frequency Old


# 1. FIXED CONSTRAINTS
# 1.1 FIX MEAN OF HIGH FREQUENCY NEW
fixed.m <- matrix(c(0, 0,
                    2, 4), nr = 2, byrow = T)

# 1.2 FIX SD OF EACH OF THE 8 DISTRIBUTIONS TO 1.0
fixed.sd <- matrix(c(1, 1,  1,  1,  1,  1,  1,  1,
                     3, 5, 13, 15, 23, 25, 33, 35), nr = 2, byrow = T)

# 1.3 FIX PROBABILITY OF HIGH FREQUENCY NEW (TO AN ARBITRARY VALUE)
fixed.p <- matrix(c(0,
                    1), nr = 2, byrow = T)

# 1.4 COMBINE FIXED CONSTRAINTS
fixed <- cbind(fixed.m, fixed.sd, fixed.p)


# 2. EQUALITY CONSTRAINTS: IDENTICAL THRESHOLDS FOR ALL SIGNALS
ident <- matrix(c(6:10,       6:10,    6:10,
                  6:10+10, 6:10+20, 6:10+30), nr = 2, byrow = T)

cat("\n-----------------------------------------------------------------\n")
cat(" MIX-2: DeCarlo (2007): Exp.1A:")
cat("\n-----------------------------------------------------------------\n")
Opti1A.Obj <- SDT.Estimate(data = data.1A, Model.Id = "MIX.2", n = n, fixed
= fixed, ident = ident, test = T)
Stat1A.Obj <- SDT.Statistics(Opti1A.Obj)
print(Stat1A.Obj)

cat("\n-----------------------------------------------------------------\n")
cat(" MIX-2: DeCarlo (2007): Exp.1B:")
cat("\n-----------------------------------------------------------------\n")
Opti1B.Obj <- SDT.Estimate(data = data.1B, Model.Id = "MIX.2", n = n, fixed
= fixed, ident = ident, test = T)
Stat1B.Obj <- SDT.Statistics(Opti1B.Obj)
print(Stat1B.Obj)

# 3. RE-FIT MODEL WITH ADDITIONAL CONSTRAINTS
# RESTRICTION: HN = HO' = LO' for Exp.1B
fixed.r <- matrix(c( 0,  0,
                     22, 32), nr = 2, byrow = T)
fixed <- cbind(fixed, fixed.r)

# RESTRICTION: HO = LO for Exp.2
```

```
ident.r <- matrix(c( 24,
                     34), nr = 2, byrow = T)
ident <- cbind(ident, ident.r)


cat("\n----------------------------------------------------------------\n")
cat(" MIX-2: DeCarlo (2007): Exp.1B:\n")
cat(" Restrictions: HO' = LO' = HN\n")
cat("               HO  = LO")
cat("\n----------------------------------------------------------------\n")
Opti1C.Obj <- SDT.Estimate(data = data.1B, Model.Id = "MIX.2", n = n, fixed
= fixed, ident = ident, test = T)
Stat1C.Obj <- SDT.Statistics(Opti1C.Obj)
print(Stat1C.Obj)

# PLOT DENSITY CURVES OF EXP.1A
SDT.Plot(Opti1A.Obj, cols = rep(1:4, each = 2), ltys = rep(c(1,2), 4),
SDT.legend = list(text = c("HF", "LF New", "HF Old", "LF Old"), cols = 1:4,
ltys = rep(1, 4)))

# PLOT ROC CURVES
# ROC with data
SDT.Plot(Opti1A.Obj, cols = c(3, 4, 6),
SDT.legend = list(text = c("LF New", "HF Old", "LF Old"), cols = c(3, 4,
6)), option = "ROC+")

# z-ROC with data
SDT.Plot(Opti1A.Obj, cols = c(3, 4, 6),
SDT.legend = list(text = c("LF New", "HF Old", "LF Old"), cols = c(3, 4,
6)), option = "zROC+")
```

The code comprises the following sections:

2. The relevant source files (`SDT-Main.R`, `SDT-Auxiliary.R`, and `SDT-MIX-PD.R`) are loaded. This section is contained within the lines:

```
# DIRECTORIES (PLEASE ADJUST TO YOUR PERSONAL SETTING)
...
source(paste(DirSource, Main, sep = ""))
```

3. The number of signals is specified

```
n <- 4
```

4. The two data sets are specified: This section is contained within the lines:

```
# DATA: DeCARLO (2007) Exp.1A: NATIVE SPEAKERS
...
             348, 356, 455, 552, 510, 1379)           # Bottom
```

5. Constraints on parameters are specified: Fixed and equality are required.

   (i) Fixed constraints:
   Two types of fixed constraints have to be specified: First, in order to fix the location of the setup both means of the first distributions (representing new high frequency words) are fixed to 0:

```
fixed.m <- matrix(c(0, 0,
                    2, 4), nr = 2, byrow = T)
```

Second, The standard deviations of all eight Gaussians are set to 1.0 (This also fixes the scale of the setup):

```
fixed.sd <- matrix(c(1, 1,  1,  1,  1,  1,  1,  1,
                     3, 5, 13, 15, 23, 25, 33, 35), nr = 2, byrow =
T)
```

Finally, the probability parameter of the first signal has to be fixed to an arbitrary value (in the present case to the value 0):

```
fixed.p <- matrix(c(0,
                    1), nr = 2, byrow = T)
```

These constraints are combined into a single matrix of fixed constraints:

```
fixed <- cbind(fixed.m, fixed.sd, fixed.p)
```

(ii) Equality constraints:
The threshold parameters for each signal assumed to be equal. This amounts to the fact that a single set of thresholds is employed:

```
ident <- matrix(c(6:10,    6:10,    6:10,
                  6:10+10, 6:10+20, 6:10+30), nr = 2, byrow = T)
```

6. The model is fit to the data of Experiment 1A and 1B using the same set of constraints and the test statistics are computed and printed:

```
Opti1A.Obj <- SDT.Estimate(data = data.1A, Model.Id = "MIX.2", n = n, fixed
= fixed, ident = ident, test = T)
Stat1A.Obj <- SDT.Statistics(Opti1A.Obj)
print(Stat1A.Obj)
...
Opti1B.Obj <- SDT.Estimate(data = data.1B, Model.Id = "MIX.2", n = n, fixed
= fixed, ident = ident, test = T)
Stat1B.Obj <- SDT.Statistics(Opti1B.Obj)
print(Stat1B.Obj)
```

7. A restricted model is specified by adding fixed and equality constraints to the existing ones and the model with these additional constraints is fitted to the data of Exp.1B:

```
# RESTRICTION: HN = HO' = LO' for Exp.1B
fixed.r <- matrix(c( 0,  0,
                     22, 32), nr = 2, byrow = T)
fixed <- cbind(fixed, fixed.r)

# RESTRICTION: HO = LO for Exp.2
ident.r <- matrix(c( 24,
                     34), nr = 2, byrow = T)
ident <- cbind(ident, ident.r)
...
Opti1C.Obj <- SDT.Estimate(data = data.1B, Model.Id = "MIX.2", n = n, fixed
= fixed, ident = ident, test = T)
Stat1C.Obj <- SDT.Statistics(Opti1C.Obj)
print(Stat1C.Obj)
```

8. Three plots are provided:
   (i) The configuration of density curves is plotted:

```
SDT.Plot(Opti1A.Obj, cols = rep(1:4, each = 2), ltys = rep(c(1,2), 4),
SDT.legend = list(text = c("HF", "LF New", "HF Old", "LF Old"), cols = 1:4,
ltys = rep(1, 4)))
```

   (ii) The ROC curves with the data are plotted:

```
SDT.Plot(Opti1A.Obj, cols = c(3, 4, 6),
SDT.legend = list(text = c("LF New", "HF Old", "LF Old"), cols = c(3, 4,
6)), option = "ROC+")
```

(iii) The *z*-ROC curves with the data are plotted:

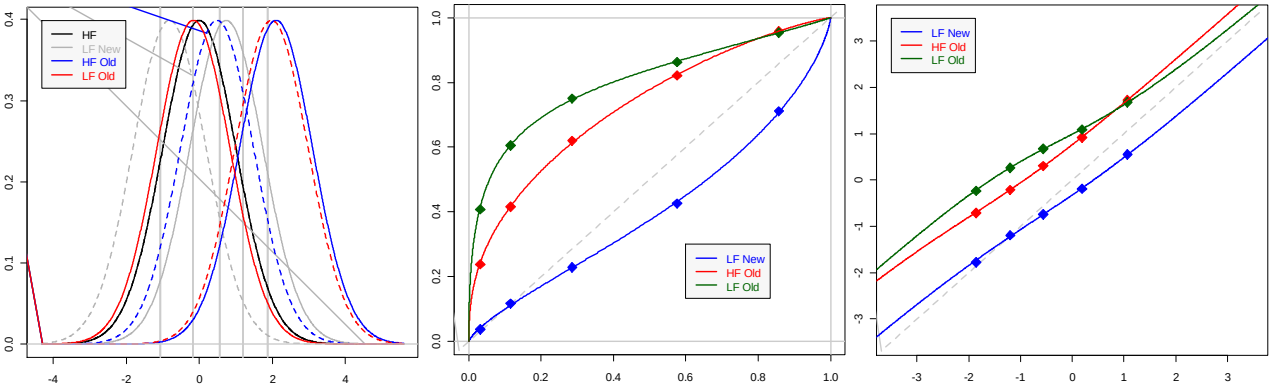
```
SDT.Plot(Opti1A.Obj, cols = c(3, 4, 6),
SDT.legend = list(text = c("LF New", "HF Old", "LF Old"), cols = c(3, 4,
6)), option = "zROC+")
```

The following excerpt shows the main portion of the output of fitting the unrestricted model to the data of Exp.2B:

```
=========================================================
 Symbolic and numeric gradients at the optimum:
 Length of difference vector  : 1.669982e-06
 Length of gradient at optimum: 0.0001051239
=========================================================
            Symbolic    Numeric Difference
t-1[1]    -1.168e-05 -1.154e-05  -1.40e-07
t-2[1]    -3.180e-06 -1.830e-06  -1.35e-06
t-3[1]    -3.293e-05 -3.308e-05   1.50e-07
t-4[1]    -6.026e-05 -6.029e-05   3.00e-08
t-5[1]     7.531e-05  7.536e-05  -6.00e-08
p[2]       5.440e-06  5.490e-06  -5.00e-08
Mean-1[2]  4.650e-06  4.600e-06   5.00e-08
Mean-2[2]  1.823e-05  1.815e-05   8.00e-08
p[3]       5.700e-07  5.000e-07   7.00e-08
Mean-1[3] -5.200e-06 -5.180e-06  -2.00e-08
Mean-2[3]  6.870e-06  7.280e-06  -4.10e-07
p[4]       1.880e-06  1.870e-06   1.00e-08
Mean-1[4] -6.700e-06 -6.600e-06  -1.00e-07
Mean-2[4]  3.250e-06  4.110e-06  -8.60e-07
=========================================================
$Model.description
[1] "Mixture Model: Mixtures of two normal distributions per signal"

$Statistics
                                        Statistic
log L                                  -19394.494
X^2                                         7.095
G^2                                         7.129
df                                          6.000
p(Y > X^2)                                  0.312
p(Y > G^2)                                  0.309
AIC                                     38816.988
BIC                                     38920.202
CAICF                                   39034.059
ICOMP                                   38842.862
ICOMP.R                                 38811.278
Free Parameters                            14.000
Length of gradient at optimum               0.000
Rank of Hessian                            14.000
Condition number of information matrix   27432.988
Rank of model matrix: t(J) * J             14.000
Condition number of model matrix         35290.090

$Free.parameters
           Value SE(delta method) SE(numerical) CFI-95(Lower) CFI-95(Upper)
t-1[1]    -0.674            0.025         0.025        -0.723        -0.626
t-2[1]     0.061            0.022         0.022         0.019         0.104
t-3[1]     0.609            0.023         0.023         0.563         0.654
t-4[1]     1.074            0.027         0.027         1.022         1.126
t-5[1]     1.591            0.036         0.036         1.521         1.661
p[2]       0.300            0.213         0.213         0.000         0.718
Mean-1[2]  0.665            0.372         0.372        -0.065         1.394
```

```
Mean-2[2] -0.417            0.195          0.195        -0.800        -0.034
p[3]       0.427            0.068          0.068         0.294         0.561
Mean-1[3]  1.947            0.204          0.204         1.547         2.347
Mean-2[3]  0.224            0.101          0.101         0.027         0.422
p[4]       0.734            0.045          0.045         0.646         0.821
Mean-1[4]  1.802            0.097          0.097         1.613         1.992
Mean-2[4] -0.035            0.144          0.144        -0.318         0.248

$Full.parametervector
          Mixture-1 Mixture-2 Mixture-3 Mixture-4
p             0.000     0.300     0.427     0.734
Mean-1        0.000     0.665     1.947     1.802
Stddev-1      1.000     1.000     1.000     1.000
Mean-2        0.000    -0.417     0.224    -0.035
Stddev-2      1.000     1.000     1.000     1.000
t-1          -0.674    -0.674    -0.674    -0.674
t-2           0.061     0.061     0.061     0.061
t-3           0.609     0.609     0.609     0.609
t-4           1.074     1.074     1.074     1.074
t-5           1.591     1.591     1.591     1.591
...
```

The fit statistics (G^2, *AIC* and *BIC*) are nearly identical to those reported by DeCarlo (2007, Table 2, on p.23: $G^2 = 7.16$, *AIC* = 38817, and *BIC* = 38920)].
The high condition numbers of the observed information matrix as well as of the model matrix indicate that the model might be too complex for the given data set.
Figure 15 displays the configuration of density curves as well as the ROC and *z*-ROC curves of Experiment 1A generated by the plot commands (cf. DeCarlo, 2007, Figure 3 on p.24).



(a) (b) (c)

Figure 15: *MIX.2 model: Plot of data and model predictions of Exp.1A of DeCarlo (2007); (a) Configuration of densities and decision bounds; (b) ROC curves, (c) z-ROC curves.*

7.2 Example 2: Fitting the MIX.2 model to source monitoring data

The file `MIX-2.1 (MIX-2 Source Monitoring, Hilford).R` contains the R-code for fitting the data from the three experiments of Hilford et al. (2002). The code looks like this:

```
# ================================================================
# MIX-2.1: Examples of the documentation of Model MIX.2:
# Modeling source monitoring data from Hilford et al. (2002)
# Date of creation: July, 2009
# Author: Siegfried Macho
# ================================================================
library(numDeriv)

# DIRECTORIES (PLEASE ADJUST TO YOUR PERSONAL SETTING)
```

```
DirSource <- "C:/Documents and Settings/macho/Statistik/_SDT-Models
(Version 2)/R-Code/"
DirSourceM <- "C:/Documents and Settings/macho/Statistik/_SDT-Models
(Version 2)/R-Code/Models/"

# FILE NAME WITH THE MODEL
Model <- "SDT-MIX-2.R"

# FILE NAMES FOR ESTIMATION
Main      <- "SDT-Main.R"
Auxiliary <- "SDT-Auxiliary.R"

# LOAD SOURCE FILES WITH SDT MODEL
source(paste(DirSourceM, Model, sep = ""))

# LOAD SOURCE FILES WITH AUXILIARY AND ESTIMATION FUNCTIONS
source(paste(DirSource, Auxiliary, sep = ""))
source(paste(DirSource, Main, sep = ""))

# DATA: HILFORD ET AL. (2002), Exp.1
data.1 <- c(264, 133, 159, 116,  80,  33,             # Male voice
             25,  62, 105, 128, 145, 294)             # Female voice


# DATA: HILFORD ET AL. (2002), Exp.2
data.2 <- c(1358, 550, 701, 585, 466,  300,           # Male
             278, 477, 598, 614, 589, 1404)           # Female


# DATA: HILFORD ET AL. (2002), Exp.3
data.3 <- c(1313, 538, 524, 529, 363,  333,           # Top
             348, 356, 455, 552, 510, 1379)           # Bottom

# 1. FIXED CONSTRAINTS
# 1.1 FIX MEANS OF THE SECOND DISTRIBUTION OF EACH MIXTURE TO 0
fixed.m <- matrix(c(0,    0,
                    4, 4+10), nr = 2, byrow = T)
# 1.2 FIX ALL SD TO 1.0
fixed.sd <- matrix(c(1,  1,    1,    1,
                     3,  5, 3+10, 5+10), nr = 2, byrow = T)

# 1.3 FUNCTIONAL CONSTRAINTS ON MEANS (MUST BE INCLUDED)
fixed.fct <- matrix(c(2,
                      2), nr = 2, byrow = T)

# 1.4 COMBINE FIXED CONSTRAINTS
fixed <- cbind(fixed.m, fixed.sd, fixed.fct)

# 2. EQUALITY CONSTRAINTS: IDENTICAL PROBABILITIES AS WELL AS IDENTICAL
THRESHOLDS
ident <- matrix(c( 1, 6:10,
                  11, 6:10+10), nr = 2, byrow = T)

# 3. FUNCTIONAL CONSTRAINT: m.A = -m.B (first distribution)
fct.m <- function(par)
{
      tindex <- 2                         # Target index: m.A
      sindex <- 2 + 10                    # Source index: m.B
```

```
      par[tindex] <- -par[sindex]

      # Define Gradient function
      gr.fct <- function(grad, par)
      {
            grad[sindex] <- grad[sindex] - grad[tindex]
            return(grad)
      }
      attr(par, "SDT.gradient") <- gr.fct             # Gradient
      return(par)
}
# 4 DEFINE FUNCTION TO COMPUTE Jacobian AND ASSIGN IT TO THE FUNCTION
J.fct <- function(par)
{
      J <- diag(length(par))
      J[2, 12] <- -1
      J
}
attr(fct.m, "SDT.Jacobian") <- J.fct                  # Jacobian matrix

cat("\n-------------------------------------------------------------\n")
cat(" MIX-2: Hilford et al. (2002): Exp.1")
cat("\n-------------------------------------------------------------\n")
Opti1.Obj <- SDT.Estimate(data = data.1, Model.Id = "MIX.2", fixed = fixed,
ident = ident, functional = fct.m, test = T)
Stat1.Obj <- SDT.Statistics(Opti1.Obj)
print(Stat1.Obj)

cat("\n-------------------------------------------------------------\n")
cat(" MIX-2: Hilford et al. (2002): Exp.2")
cat("\n-------------------------------------------------------------\n")
Opti2.Obj <- SDT.Estimate(data = data.2, Model.Id = "MIX.2", fixed = fixed,
ident = ident, functional = fct.m, test = T)
Stat2.Obj <- SDT.Statistics(Opti2.Obj)
print(Stat2.Obj)

cat("\n-------------------------------------------------------------\n")
cat(" MIX-2: Hilford et al. (2002): Exp.3")
cat("\n-------------------------------------------------------------\n")
Opti3.Obj <- SDT.Estimate(data = data.3, Model.Id = "MIX.2", fixed = fixed,
ident = ident, functional = fct.m, test = T)
Stat3.Obj <- SDT.Statistics(Opti3.Obj)
print(Stat3.Obj)

# PLOT GAUSSIAN DISTRIBUTIONS AND THRESHOLDS, AS WELL AS ROC CURVES FOR
EXP.3:
#-----------------------------------------------------------------------
# COMMENT ON THE ORDERING OF GAUSSIAN MODELS:
# The ordering is: 1. First  model of Signal 1 (m = -1.745, sd = 1.0)
#                  2. Second model of Signal 1 (m =      0, sd = 1.0)
#                  3. First  model of Signal 2 (m =  1.745, sd = 1.0)
#                  4. Second model of Signal 2 (m =      0, sd = 1.0)
#
# The second model of Signal 1 is masked be the second model of Signal 2
#-----------------------------------------------------------------------
# PLOT DENSITY CURVES
SDT.Plot(Opti3.Obj, cols = c(4, 3, 6, 3), SDT.legend = list(text =
c("left", "no info", "right"), cols = c(4, 3, 6)))
```

```
# PLOT ROC CURVES
# Plot ROC curve with data
SDT.Plot(Opti3.Obj, cols = 3, labels = c("False Alarms", "Hits"), option =
"ROC+")

# Plot z-ROC curve with data
SDT.Plot(Opti3.Obj, cols = 3, labels = c("z(FA)", "z(H)"), option =
"zROC+")
```

The code comprises the following main sections.

1. The relevant source files (`SDT-Main.R`, `SDT-Auxiliary.R`, and `SDT-MIX-PD.R`) are loaded. This section is contained within the lines:

```
# DIRECTORIES (PLEASE ADJUST TO YOUR PERSONAL SETTING)
...
source(paste(DirSource, Main, sep = ""))
```

2. The three data sets are specified: This section is contained within the lines:

```
# DATA: HILFORD ET AL. (2002), Exp.1
...
              348, 356, 455, 552, 510, 1379)           # Bottom
```

3. Constraints on parameters are specified. Note that fixed, equality, as well as functional constraints (that are neither fixed nor equality constraints) are specified.

   (i) Fixed constraints:
   Two types of fixed constraints have to be specified: First, in order to fix the scale the means of the second distributions are fixed to 0, for both signals. These distributions may be conceived of as the "middle" distribution.
   Second, The standard deviations of all four Gaussians are set to 1.0.
   Additionally to these types of fixed constraints the parameter subjected to the functional constraint (see below) is also added to the parameters of fixed constraints. Finally, these types of fixed constraints are combined into a single matrix.

   (ii) Equality constraints:
   Two types of equality constraints are specified: First, the probabilities of the employing the first distribution are equated for the two types of signals.
   Second, the threshold parameters assigned to the two types of signals are equated.

   (iii) Functional constraint:
   The first distribution associated with a signal are assumed to be the "left" distribution for the first signal and the "right" distribution for the second signal. The two distributions are assumed to be displaced from the "middle" distribution (located at 0, see above) by the same amount but with different direction. Thus $\mu_{11} = -\mu_{21}$ (the first index indicates the signal and the second one the distribution per signal). This sort of constraint is implemented by means of the function `fct.m`.

4. The function computing the Jacobian matrix of the functional constraint is computed:

```
J.fct <- function(par)
{
      J <- diag(length(par))
      J[2, 12] <- -1
      J
}
attr(fct.m, " SDT.Jacobian") <- J.fct                  # Jacobian matrix
```

The function generates and identity matrix with dimension identical to the length of the parameter vector. Then a –1 is put into the matrix: The row index of the entry corresponds to the position of the target parameters (within the full parameter vector) and the column index corresponds to the position of the source parameter.

5. The three data sets are estimated, statistics are computed, and the result is printed. For example, for modeling the data of Experiment 1, the following piece of code is relevant:

```
Opti1.Obj <- SDT.Estimate(data = data.1, Model.Id = "MIX.2", fixed = fixed,
ident = ident, functional = fct.m, test = T)
Stat1.Obj <- SDT.Statistics(Opti1.Obj)
print(Stat1.Obj)
```

The first command performs the estimation. The results of this process are stored in the object `Opti1.Obj`. Note that the number of signals ($n = 2$) was not passed to the estimation procedure because $n = 2$ is the default value.
The second command results in the computation of relevant statistics that are stored in the object `Stat1.Obj`.
Finally, the content of `Stat1.Obj` is printed.

The commands for estimating the other data sets are analogous. They main difference consists in the fact that the data sets `data.2` and `data.3` are passed (instead of `data.1`) to the estimation procedure.

6. The Gaussian density functions with the thresholds as well as ROC and z-ROC curves are plotted:
(iii) Density functions and thresholds:

```
SDT.Plot(Opti3.Obj, cols = c(4, 3, 6, 3), SDT.legend = list(text =
c("left", "no info", "right"), cols = c(4, 3, 6)))
```

As noted in the comment in the command file the second Gaussian distribution of Signal 1 is masked by the Gaussian distribution of Signal 1 due to the fact that the parameters of both distributions a specified to be equal.

(iv) ROC curves of model and data:

```
SDT.Plot(Opti3.Obj, cols = 3, labels = c("False Alarms", "Hits"), option =
"ROC+")
```

(v) *z*-ROC curves of model and data:

```
SDT.Plot(Opti3.Obj, cols = 3, labels = c("z(FA)", "z(H)"), option =
"zROC+")
```

The output for the data of Exp.1 looks like this:

```
------------------------------------------------------------------------------
 MIX-2: Hilford et al. (2002): Exp.1
------------------------------------------------------------------------------

======================================================
 Symbolic and numeric gradients at the optimum:
 Length of difference vector  : 1.625848e-07
 Length of gradient at optimum: 8.807141e-06
======================================================
           Symbolic   Numeric Difference
p[1]       3.60e-07  3.60e-07    0.0e+00
t-1[1]    -1.51e-06 -1.51e-06    1.0e-08
t-2[1]     4.62e-06  4.61e-06    1.0e-08
t-3[1]    -5.92e-06 -6.08e-06    1.6e-07
t-4[1]     4.08e-06  4.08e-06    1.0e-08
t-5[1]    -1.44e-06 -1.44e-06    1.0e-08
Mean-1[2]  3.00e-07  3.00e-07    0.0e+00
```

```
=======================================================
$Model.description
[1] "Mixture Model: Mixtures of two normal distributions per signal"

$Statistics
                                      Statistic
log L                                -2482.840
X^2                                      2.113
G^2                                      2.108
df                                       3.000
p(Y > X^2)                               0.549
p(Y > G^2)                               0.550
AIC                                   4979.680
BIC                                   5017.075
CAICF                                 5071.222
ICOMP                                 4973.280
ICOMP.R                               4970.388
Free Parameters                          7.000
Rank of Hessian                          7.000
Length of gradient at optimum            0.000
Condition number of information matrix   80.891
Rank of model matrix: t(J) * J           7.000
Condition number of model matrix       165.849

$Free.parameters
           Value SE(delta method) SE(numerical) CFI-95(Lower) CFI-95(Upper)
p[1]       0.474            0.029         0.029         0.417         0.532
t-1[1]    -1.543            0.086         0.086        -1.712        -1.374
t-2[1]    -0.835            0.056         0.056        -0.946        -0.725
t-3[1]    -0.078            0.048         0.048        -0.172         0.015
t-4[1]     0.603            0.053         0.053         0.500         0.707
t-5[1]     1.373            0.080         0.080         1.217         1.530
Mean-1[2]  1.927            0.177         0.177         1.581         2.273

$Full.parametervector
         Mixture-1 Mixture-2
p            0.474     0.474
Mean-1      -1.927     1.927
Stddev-1     1.000     1.000
Mean-2       0.000     0.000
Stddev-2     1.000     1.000
t-1         -1.543    -1.543
t-2         -0.835    -0.835
t-3         -0.078    -0.078
t-4          0.603     0.603
t-5          1.373     1.373

$Data.and.Estimates
           Observed Expected Probabilities Residuals  Std.Resid.(analytical)
Data[1, 1]      264  267.246         0.340    -0.199                 -0.662
Data[1, 2]      133  137.245         0.175    -0.362                 -0.930
Data[1, 3]      159  149.318         0.190     0.792                  1.436
Data[1, 4]      116  116.367         0.148    -0.034                 -0.055
Data[1, 5]       80   79.653         0.101     0.039                  0.080
Data[1, 6]       33   35.172         0.045    -0.366                 -0.683
Data[2, 1]       25   24.591         0.032     0.083                  0.149
Data[2, 2]       62   56.926         0.075     0.672                  1.192
Data[2, 3]      105  113.532         0.150    -0.801                 -1.440
Data[2, 4]      128  128.274         0.169    -0.024                 -0.040
Data[2, 5]      145  146.129         0.193    -0.093                 -0.239
Data[2, 6]      294  289.548         0.381     0.262                  0.848
```

Note that the result is exactly the same as that of MIX.PD model (cf. Chapter 6.1), with respect to statistics, estimated parameters, estimated standard errors, confidence intervals, and predicted frequencies and probabilities. Only the name of the parameters are different. By

consequence, the discussion of the results is analogously to that of the MIX.PD model and will not be repeated here.

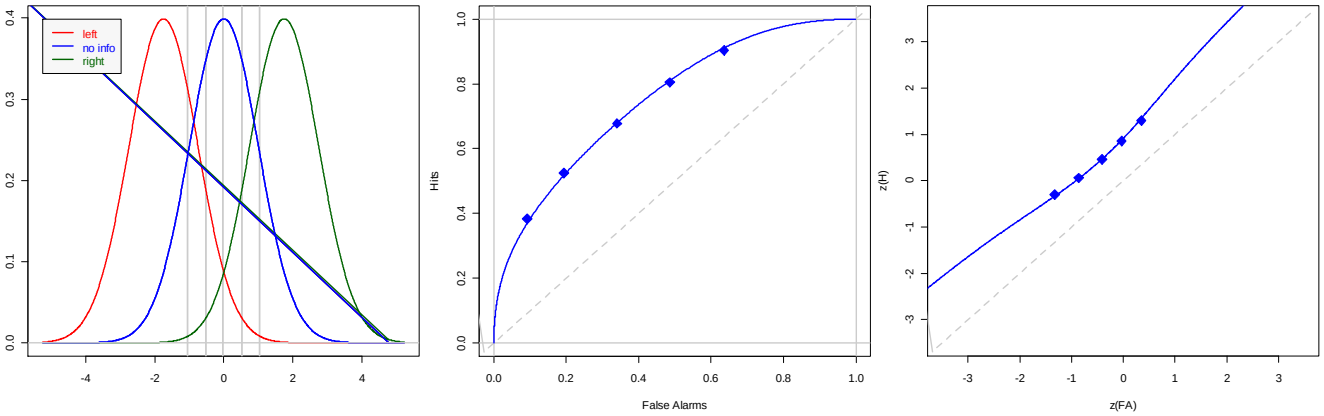
A look on the section `$Full.parametervector` reveals that the constraints on parameters have been specified properly:

(i) The *fixed constraints* on the means of the second distributions ($\mu_{12} = \mu_{22} = 0$) as well as on the standard deviations ($\sigma_{11} = \sigma_{12} = \sigma_{21} = \sigma_{22} = 1$);

(ii) The *equality constraints* on probabilities $\pi_1 = \pi_2$ as well as on thresholds for the two signals $t_{11} = t_{21}$, $t_{12} = t_{22}$, $t_{13} = t_{23}$, $t_{14} = t_{24}$, and $t_{15} = t_{25}$;

(iii) The function constraints on the means of the first distributions: $\mu_{11} = -\mu_{21}$.

The plots generated by the command SDT.Plot is shown in Figure 16.
Figure 16 displays the plots generated by the plot commands.



(a) (b) (c)

Figure 16: *MIX.2 model: Plot of data and model predictions of Exp.3 of Hilford et al (2003): (a) Configuration of densities and decision bounds; (b) ROC curves, (c) z-ROC curves.*

Comment:
The order of the density functions plotted is different from that of the MIX.PD model (cf. Figure 13). This is due to the parameterization: The first signal is represented by the left (red) and middle (blue) distribution whereas the second signal is represented by the right (green) and middle distribution.

7.3 Example 3: Fitting the MIX.2 model to data on associative recognition

The file `MIX-2.2 (MIX-2 Associative Recognition, Exp.2).R` contains the R-code for fitting the model to the data from Experiment 2 of Kelley and Wixted (2001). For this example, the constraints are specified in such a way that the outcome of the MIX.2 model can be compared directly with that of MIX.PD shown above.
The relevant code, that is, the specification of the data, the specification of restrictions, and the commands for fitting the model and computing test statistics are shown below (The file contains additional code that lists the names of the parameter in the order they occur within the parameter vector. This may be helpful for the specification of constraints. This piece of code is not shown here).

1. Specification of the data:
   The data are presented in the same order as for the MIX.PD model (otherwise the ROC plots would not be correct)

```
  data <- c(879, 553, 345,  61,  54,  28, # Exp.2: Lure items
            179, 109,  82,  45,  28,  37, # Exp.2: Weak   Items, rearranged
             81,  69,  62,  31,  37, 200, # Exp.2: Weak   Items, intact
            215,  82,  60,  31,  31,  61, # Exp.2: Strong Items, rearranged
             21,  24,  33,  21,  33, 348) # Exp.2: Strong Items, intact
```

2. Specification of constraints:
   The hard work when using the MIX.2 model consists in specifying the relevant constraints on parameters (As already noted, without imposing constraints on parameters the model is not identified). We begin with the specification of fixed constraints:

```
# 1. SPECIFY FIXED CONSTRAINTS
# 1.1 FIX PARAMETERS FOR THE LURE DISTRIBUTION: p.1 = 0, m.11 = m.12 = 0,
sd.11 = sd.12 = 1
#                      p.1 m.11 m.12 sd.11 sd.12
fixed.lure <- matrix(c( 0,   0,   0,   1,   1,
                        1,   2,   4,   3,   5), nr = 2, byrow = T)

# 1.2 FUNCTIONAL CONSTRAINTS ON MEANS (SEE BELOW)
#                     m.31 m.32 m.41 m.51 m.52
fixed.fct <- matrix(c(   0,   0,   0,   0,   0,
                        22,  24,  32,  42,  44), nr = 2, byrow = T)


# 1.3 COMBINE FIXED CONSTRAINTS
fixed <- cbind(fixed.lure, fixed.fct)
```

   The fixed constraints consist of two types of constraints:

   (i) The parameters of the model used for modeling the data from the lure signal are fixed completely (cf. the matrix `fixed.lure`). Specifically, the two means are fixed to 0 and the standard distributions of the two Gaussians (that are assigned to each signal) were fixed to 1.0. In addition, the probability parameter was fixed to 0 (Note that it may be fixed to any arbitrary value because the two mixing distributions are equal).
   (ii) The parameters that are subjected to functional constraints are declared as fixed (cf. the matrix `fixed.fct`).

   The two matrices of fixed constraints are then combined into a single matrix `fixed`.

   Second, equality constraints were specified:

```
# 2. SPECIFY EQUALITY CONSTRAINTS: IDENTICAL PROBABILITIES AND THRESHOLD
IDENTICAL
# 2.1 IDENTICAL THRESHOLDS
ident.th <- matrix(c(6:10,    6:10,    6:10,    6:10,
                  6:10+10, 6:10+20, 6:10+30, 6:10+40), nr = 2, byrow = T)

# 2.2 IDENTICAL SD FOR STRONG SIGNALS AND IDENTICAL SD FOR WEAK SIGNALS
ident.sd <- matrix(c(   13,   13,    13, 13+20, 13+20,  13+20,
                     13+10,   15, 15+10, 13+30, 15+20,  15+30), nr = 2,
byrow = T)

# 2.3 COMBINE EQUALITY CONSTRAINTS
ident <- cbind(ident.th, ident.sd)
```

   There are also two different kinds of equality constraints:

   (i) The standard deviations of the four distributions for modeling the weak signals are equated and the same is done for the four distributions for modeling the strong signals (cf. the matrix `ident.sd`).
   (ii) The thresholds associated with different signals are equated (cf. the matrix `ident.th`).

The two matrices of equality constraints are then combined into a single matrix `ident`.

Third, functional constraints are specified. This piece of the code is certainly the most intricate one:

```
# 3. SPECIFY FUNCTIONAL CONSTRAINTS
fct.m <- function(par)
{
      # Positions of target means
      m.31 <- 22                # Target index: m.31
      m.32 <- 24                # Target index: m.32
      m.41 <- 32                # Target index: m.41
      m.51 <- 42                # Target index: m.51
      m.52 <- 44                # Target index: m.52

      # Positions of source means
      m.21 <- 12                # Source index: m.21
      m.22 <- 14                # Source index: m.22
      m.42 <- 34                # Source index: m.42

      # Constraints:
      par[m.32] <- par[m.22]    # Fix "middle" distributions, weak   pair
      par[m.52] <- par[m.42]    # Fix "middle" distributions, strong pair

      par[m.31] <-  par[m.22] + par[m.22] - par[m.21]      # Right =  middle +
d'.ass (weak)
      par[m.41] <-  par[m.42] + par[m.21] - par[m.22]      # Left  =  middle -
d'.ass (strong)
      par[m.51] <-  par[m.42] - par[m.21] + par[m.22]      # Right =  middle +
d'.ass  (strong)

      # Define Gradient function
      gr.fct <- function(grad, par)
      {
            grad[m.21] <- grad[m.21] -   grad[m.31] + grad[m.41] - grad[m.51]
            grad[m.22] <- grad[m.22] + 2*grad[m.31] + grad[m.32] - grad[m.41] +
grad[m.51]
            grad[m.42] <- grad[m.42] +   grad[m.52] + grad[m.41] + grad[m.51]

            return(grad)
      }
      attr(par, "SDT.gradient") <- gr.fct                  # Gradient
      return(par)
}
```

The following steps are performed within the function:

(i) The positions of the target and source parameters are specified. The names of the indices indicate the parameter: `m.ij` indicates the *j*-th mean of the *i*-th signal, for example `m.31` denotes the position of the mean of the first distribution for the third signal.
A parameter may function either as a target or as source of a constraint but not as both.

(ii) It was assumed that the second distributions assigned to a signal always function as the "middle" distribution (as in the MIX.PD model). In addition the "middle" distributions are the same for intact and rearranged items. Thus the means of the second distribution of the first and second signal must be equated, and, similarly, the means of the third and fourth distribution must be identical (Note that the standard deviations of the distributions for weak and strong pairs were equated by means of equality constraints). This is performed by means of the two commands:

```
par[m.32] <- par[m.22]     # Fix "middle" distributions, weak   pair
par[m.52] <- par[m.42]     # Fix "middle" distributions, strong pair
```

(iii) It was further assumed that the first distribution assigned to a signal functions either as the "left" distribution or as the "right" distribution. Specifically, for *weak items* the first distribution assigned to the first signal functions as the "left" and the first distribution assigned to the second signal works as the "right" distribution. Similarly, for *strong items* the first distribution assigned to the third signal functions as the "left" and the first distribution assigned to the forth signal works as the "right" distribution.

Now, the restriction that the displacement d' of the "left" and "right" distributions form the "middle" distributions are the same has to be implemented for strong and weak pairs. The displacement d'_{left} for the weak items is given by: $d'_{left} = \mu_{11} - \mu_{12}$. This quantity has to be subtracted (in case of "right" distributions) or added (in case of a "left" distribution) to the means of the "middle" distributions. This is done by the three commands:

```
par[m.31] <- par[m.22] + par[m.22] - par[m.21] # Right =  middle + d'.ass (weak)
par[m.41] <- par[m.42] + par[m.21] - par[m.22] # Left  =  middle - d'.ass (strong)
par[m.51] <- par[m.42] - par[m.21] + par[m.22] # Right =  middle + d'.ass (strong)
```

The first command specifies the mean of the "right" distribution for the weak items, the second one determines the mean of "left" distribution of the strong items and the final one provides the mean of the "right" distribution of the strong items.

(iv) The gradient must be adjusted. Specifically, the partial derivatives of the log likelyhood with respect to the source parameters `m.21`, `m.22`, and `m.42` must be adapted. In the present case this is very simple: One has to add to the existing derivative the derivatives of those target parameters that are influenced by the respective source parameter. If the source parameter is added to the target the sign of the derivative is positive, and if the source parameter is subtracted, the sign of the respective derivative is negative.

Take, for example, parameter `m.22`. The partial derivative is adjusted in the following way:

```
grad[m.22] <- grad[m.22] + 2*grad[m.31] + grad[m.32] - grad[m.41] + grad[m.51]
```

The derivative `grad[m.32]` is added because of the restriction: `par[m.32] <-` **`par[m.22].`**
The derivative `2*grad[m.31]` is added due to the restriction: `par[m.31] <-` **`par[m.22] + par[m.22]`** `- par[m.21].` (note that `m.22` enters twice).
The derivative `grad[m.41]` is subtracted because of the restriction: `par[m.41] <- par[m.42] + par[m.21]` **`- par[m.22].`**
Finally, the derivative `grad[m.51]` is added due to the restriction: `par[m.51] <- par[m.42] - par[m.21]` **`+ par[m.22].`**

(v) The function computing the Jacobian matrix of the functional constraints is specified:

```
# 4. SPECIFY JACOBIAN ASSOCIATED WITH FUNCTIONAL CONSTRAINTS
J.fct <- function(par)
{
      # Positions of target means
      m.31 <- 22                # Target index: m.31
      m.32 <- 24                # Target index: m.32
      m.41 <- 32                # Target index: m.41
      m.51 <- 42                # Target index: m.51
      m.52 <- 44                # Target index: m.52

      # Positions of source means
      m.21 <- 12                # Source index: m.21
      m.22 <- 14                # Source index: m.22
```

```
       m.42 <- 34                # Source index: m.42

       J <- diag(length(par))

       # Implement Constraints:
       J[m.32, m.22] <-  1
       J[m.52, m.42] <-  1

       J[m.31, m.22] <-  2
       J[m.31, m.21] <- -1

       J[m.41, m.42] <-  1
       J[m.41, m.21] <-  1
       J[m.41, m.22] <- -1

       J[m.51, m.42] <-  1
       J[m.51, m.21] <- -1
       J[m.51, m.22] <-  1
       J
}

attr(fct.m, "SDT.Jacobian") <- J.fct                    # Jacobian matrix
```

As for the previous examples the positions of the target parameters make up the row indices and the positions of the source parameters make up the column indices of the relevant entries.

3. Estimation and computation of test statistics: After the specification of the constraints it is quite simple to estimate the model and compute the statistic. This is performed as usual, by the following pair of commands:

```
Opti.Obj <- SDT.Estimate(data = data, Model.Id = "MIX.2", n = n, fixed =
fixed, ident = ident, functional = fct.m, test = T)
Stat.Obj <- SDT.Statistics(Opti.Obj)
```

The first command results in the estimation of the model, with the results contained in the object `Opti.Obj`. The second command leads to the computation of the test statistics that is stored in the object `Stat.Obj`.

4. Plots of the density and ROC curves with data are generated by three calls of the plot function `SDT.Plot`: (the resulting plots are identical to that of Figure 14).

```
# PLOT GAUSSIAN DISTRIBUTIONS AND THRESHOLDS:
# PLOT DENSITY CURVES
SDT.Plot(Opti.Obj, cols = c(6, 6, 3, 3, 3, 3, 4, 4, 4, 4), SDT.legend =
list(text = c("New", paste("Weak", 1:3),paste("Strong", 1:3)), cols =c(6,
3, 3, 3, 4, 4, 4)))

# PLOT ROC CURVES
# Plot ROC curve with data
SDT.Plot(Opti.Obj, cols = c(3, 3, 4, 4), ltys = c(2, 1, 2, 1), labels =
c("False Alarms", "Hits"), option = "ROC+",
SDT.legend = list(text = c("weak rearranged", "weak intact","strong
rearranged", "strong intact"), cols = c(3, 3, 4, 4), ltys = c(2, 1, 2, 1)))

# Plot z-ROC curve with data
SDT.Plot(Opti.Obj, cols = c(3, 3, 4, 4), ltys= c(2, 1, 2, 1), labels =
c("z(FA)", "z(H)"), option = "zROC+",
SDT.legend = list(text = c("weak rearranged", "weak intact","strong
rearranged", "strong intact"), cols = c(3, 3, 4, 4), ltys = c(2, 1, 2, 1)))
```

Due to the setting of the test flag of the function `SDT.Estimate()` the following table is printed:

```
=======================================================
 Symbolic and numeric gradients at the optimum:
 Length of difference vector  : 2.588069e-07
 Length of gradient at optimum: 4.976604e-05
=======================================================
              Symbolic    Numeric Difference
t-1[1]       2.986e-05  3.010e-05   -2.4e-07
t-2[1]      -2.668e-05 -2.672e-05    4.0e-08
t-3[1]       1.603e-05  1.600e-05    3.0e-08
t-4[1]      -1.860e-05 -1.860e-05    0.0e+00
t-5[1]       1.347e-05  1.348e-05   -1.0e-08
p[2]         7.500e-07  7.300e-07    2.0e-08
Mean-1[2]    2.230e-06  2.240e-06   -1.0e-08
Stddev-1[2]  2.500e-06  2.490e-06    1.0e-08
Mean-2[2]   -8.310e-06 -8.390e-06    7.0e-08
p[3]        -1.870e-06 -1.880e-06    1.0e-08
p[4]         3.200e-07  3.600e-07   -4.0e-08
Stddev-1[4]  4.000e-07  4.200e-07   -2.0e-08
Mean-2[4]   -1.970e-06 -1.970e-06    0.0e+00
p[5]        -5.500e-07 -5.300e-07   -2.0e-08
=======================================================
```

In the present case, the table showing a comparison of the numeric and analytic gradient is very important since it enables one to check whether the gradient vector was adjusted correctly. In case of an incorrect adjustment of gradient a major difference between the numeric and the analytic gradient would be observed.

Assume, for the instance, that the derivative of `m.22` was adjusted (wrongly) in the following way (instead of `2*grad[m.31]` the value enters only once):

```
grad[m.22] <- grad[m.22] + grad[m.31] + grad[m.32] - grad[m.41] + grad[m.51]
```

This would result in the following output:

```
=======================================================
 Symbolic and numeric gradients at the optimum:
 Length of difference vector  : 0.3612789
 Length of gradient at optimum: 0.004766168
=======================================================
               Symbolic     Numeric  Difference
t-1[1]       0.00022215  0.00022219 -0.00000005
t-2[1]       0.00188639  0.00188639  0.00000001
t-3[1]       0.00050008  0.00050008  0.00000001
t-4[1]       0.00070468  0.00070470 -0.00000002
t-5[1]      -0.00318094 -0.00318094  0.00000001
p[2]         0.00032329  0.00032328  0.00000001
Mean-1[2]   -0.00159187 -0.00159186 -0.00000001
Stddev-1[2]  0.00090994  0.00090993  0.00000001
Mean-2[2]    0.00072353 -0.36055539  0.36127892
p[3]         0.00083546  0.00083530  0.00000015
p[4]        -0.00037083 -0.00037079 -0.00000004
Stddev-1[4] -0.00097489 -0.00097487 -0.00000002
Mean-2[4]    0.00142946  0.00142947 -0.00000002
p[5]         0.00065129  0.00065130 -0.00000001
=======================================================
```

Inspection of the table reveals that the specification of the derivative of `m.22` (= `Mean-2[2]`) was not correct.

Printing the contents of `Stat.Obj` (with all constraints and derivatives specified correctly) leads to the following output:

```
$Model.description
```

```
[1] "Mixture Model: Mixtures of two normal distributions per signal"

$Statistics
                                      Statistic
log L                                 -5249.008
X^2                                      13.004
G^2                                      12.867
df                                       11.000
p(Y > X^2)                                0.293
p(Y > G^2)                                0.302
AIC                                   10526.017
BIC                                   10613.562
CAICF                                 10709.538
ICOMP                                 10545.648
ICOMP.R                               10515.016
Free Parameters                          14.000
Rank of Hessian                          14.000
Length of gradient at optimum             0.000
Condition number of information matrix 12120.751
Rank of model matrix: t(J) * J           14.000
Condition number of model matrix       6796.958

$Free.parameters
             Value SE(delta method) SE(numerical) CFI-95(Lower) CFI-95(Upper)
t-1[1]      -0.108            0.029         0.029        -0.164        -0.052
t-2[1]       0.675            0.029         0.029         0.618         0.733
t-3[1]       1.402            0.039         0.039         1.326         1.478
t-4[1]       1.746            0.047         0.047         1.654         1.838
t-5[1]       2.187            0.063         0.063         2.065         2.310
p[2]         0.099            0.066         0.066         0.000         0.229
Mean-1[2]   -2.211            0.295         0.295        -2.790        -1.632
Stddev-1[2]  1.238            0.078         0.078         1.084         1.391
Mean-2[2]    0.509            0.123         0.123         0.268         0.749
p[3]         0.461            0.058         0.058         0.348         0.575
p[4]         0.306            0.214         0.214         0.000         0.726
Stddev-1[4]  1.475            0.308         0.308         0.871         2.080
Mean-2[4]    0.850            0.543         0.543        -0.215         1.915
p[5]         0.843            0.141         0.141         0.566         1.000

$Full.parametervector
         Mixture-1 Mixture-2 Mixture-3 Mixture-4 Mixture-5
p            0.000     0.099     0.461     0.306     0.843
Mean-1       0.000    -2.211     3.228    -1.870     3.569
Stddev-1     1.000     1.238     1.238     1.475     1.475
Mean-2       0.000     0.509     0.509     0.850     0.850
Stddev-2     1.000     1.238     1.238     1.475     1.475
t-1         -0.108    -0.108    -0.108    -0.108    -0.108
t-2          0.675     0.675     0.675     0.675     0.675
t-3          1.402     1.402     1.402     1.402     1.402
t-4          1.746     1.746     1.746     1.746     1.746
t-5          2.187     2.187     2.187     2.187     2.187

$Data.and.Estimates
...
```

A comparison of test statistics, estimated parameters and standard errors with those of the model MIX.PD in Chapter 6.2 reveals that the test statistics and the comparable parameters as well as estimated standard errors are identical. For example, the mean of the "middle" distribution for weak items was estimated as 0.509 in both cases with an estimated standard error of 0.123.

We can also check, whether the estimated displacement d' of the "left" and "right" distribution, respectively, from the "middle" one was identical to that of MIX.PD. For the latter the estimated value was d' = ±2.720 (cf. Chapter 6.2). In the present case we have:

```
m.21 - m.22:   –2.211 – 0.509  =  –2.720
m.31 - m.32:    3.228 – 0.509  =   2.719
m.41 - m.42:   –1.870 – 0.850  =  –2.720
m.51 - m.52:    3.569 – 0.850  =   2.719
```

Thus, up to negligible rounding errors the displacements are exactly the same for both models. The computation also demonstrates that the functional constraints on parameters were specified correctly.

Comment:
Contrary to the example in Chapter 7.1, the condition number of the Jacobian of the MIX-2 model does not correspond exactly to that of the MIX-PD model (cf. Chapter 6.2). The reason for this divergence is not clear to me. However, the numerically and analytically computed Jacobians are identical for the present example.

7.4 Source files containing further examples

The package comprises additional source files (not mentioned above):

- ❒ `MIX-2.3 (MIX-2 Parameter Information).R`
- ❒ `MIX-2.4 (MIX-2 Process Dissociation, Yonelinas, 1994).R`
- ❒ `MIX-2.5.2 (MIX-2 Mirror effect,Arndt & Reder, 2002, Exp.1).R`
- ❒ `MIX-2.6.1 (MIX-2 Plot, Source Monitoring).R`
- ❒ `MIX-2.6.2 (MIX-2 Plot, Associative Recognition).R`
- ❒ `MIX-2.6.3 (MIX-2 Plot, Process Dissociation).R`
- ❒ `MIX-2.6.4 (MIX-2 Plot, Mirror Effect).R`
- ❒ `MIX-2.7 (MIX-2 SDT Model).R`

The first file demonstrates the printing or parameter information.
The second file contains the code for modeling the data of Yonelinas (1994) with the models proposed by DeCarlo (2008).
The third file contains the code for mixture modeling the data on the mirror effect of Arndt and Reder (2002, Exp.1). The fit of the model does not correspond to that of DeCarlo (2007, Table 6, on p.28). The reasons for the divergence is not clear. Using the values reported by DeCarlo (2007, Table 7, on p. 29) as starting values for the estimation procedure leads to the same result as using the default starting values. The same model implemented by means of Excel leads to exactly the same fit results as that provided by R (and are thus different from those of DeCarlo).
The next four files (`MIX-2.6.1` to `MIX-2.6.4`) illustrate the usage of the plot function using plotting objects instead of the estimation object (cf. Chapter 2.5).
The last file (`MIX-2.7`) demonstrates the fitting of the standard dual process model to old-new recognition data. Two models are fitted: (a) The standard model assuming that in case of an old items the new distribution is used with a certain probability whereas the old distribution is used with the complementary probability; and (b) the extended model assuming that instead of using the new distribution another distribution whose mean and standard deviation are estimated is used with a certain probability (and the old distribution is used with the complementary probability) [**Comment:** For the extended model the condition number of the observed information matrix is relatively high, indicating problems of identification].

8. Working Examples: Dual process signal detection model (DPSDT)

The present chapter provides examples for fitting the dual process signal detection model (for a description of the model, cf. Chapter 3.5). The application of the model is illustrated with the same set of data that were used for the Gaussian mixture model:

(i) Source monitoring data from Hilford et al. (2002)
(ii) Data on associative recognition from Kelley and Wixted (2001).

8.1 Example 1: Fitting the DPSDT model to source monitoring data

The file `DPSDT-1 (Source Monitoring, Hilford).R` contains the R-code for fitting the data from the three experiments of Hilford et al. (2002). The code looks like this:

```
# ============================================================
# Examples of the documentation of Model DPSDT:
# DPSDT-1: Source monitoring with two signals
#
# Data from Hilford et al. (2002)
# Date of creation: November, 2013
# Author: Siegfried Macho
# ============================================================
library(numDeriv)

# DIRECTORIES (PLEASE ADJUST TO YOUR PERSONAL SETTING)
DirSource <- "C:/Documents and Settings/macho/Statistik/_SDT-Models
(Version 2)/R-Code/"
DirSourceM <- "C:/Documents and Settings/macho/Statistik/_SDT-Models
(Version 2)/R-Code/Models/"

# FILE NAME WITH THE MODEL
Model <- "SDT-DPSDT.R"

# FILE NAMES FOR ESTIMATION
Main      <- "SDT-Main.R"
Auxiliary <- "SDT-Auxiliary.R"

# LOAD SOURCE FILES WITH SDT MODEL
source(paste(DirSourceM, Model, sep = ""))

# LOAD SOURCE FILES WITH AUXILIARY AND ESTIMATION FUNCTIONS
source(paste(DirSource, Auxiliary, sep = ""))
source(paste(DirSource, Main, sep = ""))


# DATA: HILFORD ET AL. (2002), Exp.1
data.1 <- c(264, 133, 159, 116,  80,  33,       # Male voice
             25,  62, 105, 128, 145, 294)       # Female voice


# DATA: HILFORD ET AL. (2002), Exp.2
data.2 <- c(1358, 550, 701, 585, 466,  300,     # Male
             278, 477, 598, 614, 589, 1404)     # Female


# DATA: HILFORD ET AL. (2002), Exp.3
data.3 <- c(1313, 538, 524, 529, 363,  333,     # Top
             348, 356, 455, 552, 510, 1379)     # Bottom

cat("\n-------------------------------------------------------\n")
```

```
cat(" DPSDT: Hilford et al. (2002), Exp.1: Identical recollection
probabilities")
cat("\n------------------------------------------------------------\n")
cfg <- list(n.sdt = 2, restriction = "standard-2-eq")
Opti1.Obj <- SDT.Estimate(data = data.1, n = cfg, Model.Id = "DPSDT", test
= T)
Stat1.Obj <- SDT.Statistics(Opti1.Obj)
print(Stat1.Obj)

cat("\n------------------------------------------------------------\n")
cat(" DPSDT: Hilford et al. (2002), Exp.2: Identical recollection
probabilities")
cat("\n------------------------------------------------------------\n")
Opti2.Obj <- SDT.Estimate(data = data.2, n = cfg, Model.Id = "DPSDT", test
= T)
Stat2.Obj <- SDT.Statistics(Opti2.Obj)
print(Stat2.Obj)

cat("\n------------------------------------------------------------\n")
cat(" DPSDT: Hilford et al. (2002), Exp.3: Identical recollection
probabilities")
cat("\n------------------------------------------------------------\n")
Opti3.Obj <- SDT.Estimate(data = data.3, n = cfg, Model.Id = "DPSDT", test
= T)
Stat3.Obj <- SDT.Statistics(Opti3.Obj)
print(Stat3.Obj)
```

The code consists of three sections.

1. The relevant source files (`SDT-Main.R`, `SDT-Auxiliary.R`, and `SDT-DPSDT.R`) are loaded. This section is contained within the lines:

```
# DIRECTORIES (PLEASE ADJUST TO YOUR PERSONAL SETTING)
...
source(paste(DirSource, Main, sep = ""))
```

2. The three data sets are specified: This section is contained within the lines:

```
# DATA: HILFORD ET AL. (2002), Exp.1
...
             348, 356, 455, 552, 510, 1379)           # Bottom
```

3. The three data sets are estimated, statistics are computed and the result is printed. For example, for modeling the data of Experiment 1, the following piece of code is relevant:

```
cfg <- list(n.sdt = 2, restriction = "standard-2-eq")
Opti1.Obj <- SDT.Estimate(data = data.1, n = cfg, Model.Id = "DPSDT", test
= T)
Stat1.Obj <- SDT.Statistics(Opti1.Obj)
print(Stat1.Obj)
```

- ❑ The command in the first row specifies the configuration information consisting of the number of signals and the type of restrictions that are set (cf. Chapter 3.5).
- ❑ The second command performs the estimation. The results of this process are stored in the object `Opti1.Obj`.
- ❑ The third command performs the computation of relevant statistics that are stored in the object `Stat1.Obj`.
- ❑ Finally, the content of `Stat1.Obj` is printed.

The commands for estimating the other data sets are analogously. They main difference consists in the fact that the data sets `data.2` and `data.3` are passed (instead of `data.1`) to the estimation procedure.

The output for the data of Exp.1 looks like this:

```
------------------------------------------------------------------------------
 DPSDT: Hilford et al. (2002), Exp.1: Identical recollection probabilities---------
------------------------------------------------------------------------

=========================================================
 Symbolic and numeric gradients at the optimum:
 Length of difference vector  : 2.131116e-08
 Length of gradient at optimum: 7.987202e-06
=========================================================
                 Symbolic   Numeric Difference
Recollection[1]  5.94e-06  5.94e-06      0e+00
t-1[1]           6.70e-07  6.80e-07     -1e-08
t-2[1]          -3.78e-06 -3.79e-06      0e+00
t-3[1]           3.03e-06  3.03e-06     -1e-08
t-4[1]          -9.90e-07 -9.90e-07      0e+00
t-5[1]           1.60e-07  1.70e-07     -1e-08
Mean[2]         -1.88e-06 -1.86e-06     -1e-08
=========================================================

$Model.description
[1] "Dual Process SDT model: Number of Gaussian models: <2> Type of restrictions:
<standard-2-eq>"

$Statistics
                                      Statistic
log L                                 -2482.375
X^2                                       1.177
G^2                                       1.178
df                                        3.000
p(Y > X^2)                                0.759
p(Y > G^2)                                0.758
AIC                                    4978.750
BIC                                    5016.145
CAICF                                  5070.960
ICOMP                                  4976.811
ICOMP.R                                4973.882
Free Parameters                           7.000
Rank of Hessian                           7.000
Length of gradient at optimum             0.001
Condition number of information matrix   243.180
Rank of model matrix: t(J) * J            7.000
Condition number of model matrix        521.792

$Free.parameters
                 Value SE(delta) SE(numerical) CFI-95(Lower) CFI-95(Upper)
Recollection[1]  0.211     0.038         0.038         0.136         0.287
t-1[1]          -0.970     0.149         0.149        -1.263        -0.678
t-2[1]          -0.298     0.093         0.093        -0.479        -0.116
t-3[1]           0.330     0.075         0.075         0.184         0.476
t-4[1]           0.892     0.065         0.065         0.764         1.020
t-5[1]           1.580     0.065         0.065         1.453         1.707
Mean[2]          0.790     0.131         0.131         0.534         1.046

$Full.parametervector
                 DPSDT.1 DPSDT.2
Recollection-pos   1.000   6.000
Recollection       0.211   0.211
Mean               0.000   0.790
Stddev             1.000   1.000
t-1               -0.970  -0.970
```

```
t-2               -0.298  -0.298
t-3                0.330   0.330
t-4                0.892   0.892
t-5                1.580   1.580

$Data.and.Estimates
           Observed Expected Probabilities Residuals  Std.Resid.(analytical)
Signal 1 [1]      264  268.579         0.342    -0.279                 -0.831
Signal 1 [2]      133  134.338         0.171    -0.115                 -0.330
Signal 1 [3]      159  152.549         0.194     0.522                  0.928
Signal 1 [4]      116  114.293         0.146     0.160                  0.253
Signal 1 [5]       80   79.919         0.102     0.009                  0.017
Signal 1 [6]       33   35.322         0.045    -0.391                 -0.773
Signal 2 [1]       25   23.451         0.031     0.320                  0.732
Signal 2 [2]       62   59.360         0.078     0.343                  0.631
Signal 2 [3]      105  110.402         0.145    -0.514                 -0.855
Signal 2 [4]      128  130.499         0.172    -0.219                 -0.361
Signal 2 [5]      145  146.381         0.193    -0.114                 -0.279
Signal 2 [6]      294  288.906         0.381     0.300                  0.845
```

The table at the beginning of the output is presented because the flag `test=T` was set in the estimation procedure. By consequence, the numeric gradient at the optimum is computed using the function `grad()` from the R package `numDeriv` that was includes at the beginning of the code file.
The numeric and symbolic gradients are close together, and, in addition, the length of the gradient at the optimum is close to zero, indicating that a stationary point was found.

The model identification string indicates that the correct model was used and the statistics sections provides the relevant general statistical information (for details, cf. Chapter 2.2).

The next section in the output lists the free parameters and their estimated values the estimated standard errors (SE), as well as the lower and upper 95% confidence limits. In the present case, only π_1 (the recollection parameter of the first model) and μ_2 (the mean parameter of the second model) as well as the thresholds were estimated.
The standard errors (SE) of parameters are estimated in two ways:

1. Using the Hessian matrix that is provided by the estimation procedure together with the delta method.
2. Computation of the Hessian, using the procedure `hessian()` from the R package `numDeriv`.

The resulting estimates are shown in the second and first column from the right. Conflicting values indicate a problem of the estimation. The may happen, for instance, if one of the probability parameters is close to the limit (i.e. near 0.0 or 1.0).
In the present case, the estimates resulting from the two methods are identical. So there seems to be no problem.
The computation of the limits of the confidence intervals is based on standard errors computed by means of the delta method.

The next section presents the full parameter vector. The first line labeled `Recollection-pos` shows the response categories that is selected in case of recollection. In the present case, the first response category is selected for the first signal and the last response category for the second signal. Inspection reveals that the intended equality constraints on recollection parameter were set correctly.

The final section provides the observed data, the estimated model frequencies and probabilities as well as the Pearson and standardized residuals (cf. Chapter 2.2).

8.2 Example 2: Fitting the DPSDT model to associative recognition data

The files:

```
DPSDT-2.1 (Associative Recognition, Kelly & Wixted, Exp.1).R
DPSDT-2.2 (Associative Recognition, Kelly & Wixted, Exp.2).R
DPSDT-2.3 (Associative Recognition, Kelly & Wixted, Exp.3).R
```

contain R-code for fitting the DPDST model to the data from the three experiments of Kelley & Wixted (2001) on associative recognition. The following excerpt from `DPSDT-2.2 (Associative Recognition, Kelly & Wixted, Exp.2).R` contains the portion of the code for fitting one model to the data of Experiment 2:

```
# ============================================================
# Examples of the documentation of Model DPSDT:
# DPSDT-2.2: Modeling Associative Recognition
#
# Data from Kelley and Wixted, Exp.2
# Date of creation: November, 2013
# Author: Siegfried Macho
# ============================================================...
...[Code for loading relevant source files]

# DATA OF KELLEY & WIXTED (2001), Experiment 2
data <- c(879, 553, 345,  61,  54,  28,   # Exp.2: Lure items
          179, 109,  82,  45,  28,  37,   # Exp.2: Weak   Items, rearranged
           81,  69,  62,  31,  37, 200,   # Exp.2: Weak   Items, intact
          215,  82,  60,  31,  31,  61,   # Exp.2: Strong Items, rearranged
           21,  24,  33,  21,  33, 348)   # Exp.2: Strong Items, intact

cfg <- list(n.sdt = 5, restriction = "standard-lure-pairs-eq", rec.pos =
c(1, 1, 6, 1, 6))
Opti1.Obj <- SDT.Estimate(data = data, Model.Id = "DPSDT", n = cfg, test =
T)
Stat1.Obj <- SDT.Statistics(Opti1.Obj)
print(Stat1.Obj)

... [Code for fitting other models]
```

The code comprises four principal sections.

1. The relevant source files (`SDT-Main.R`, `SDT-Auxiliary.R`, and `SDT-DPSDT.R`) are loaded. This section was omitted:

```
... [Code for loading relevant source files]
```

2. The data sets for the five different types of signals are specified:

```
# DATA OF KELLEY & WIXTED (2001), Experiment 2
data <- c(879, 553, 345,  61,  54,  28,   # Exp.2: Lure items
          179, 109,  82,  45,  28,  37,   # Exp.2: Weak   Items, rearranged
           81,  69,  62,  31,  37, 200,   # Exp.2: Weak   Items, intact
          215,  82,  60,  31,  31,  61,   # Exp.2: Strong Items, rearranged
           21,  24,  33,  21,  33, 348)   # Exp.2: Strong Items, intact
```

Note that at the beginning the responses for the lure items is given. The data for lure items have always to be presented before the item pairs.

3. The configuration list is specified:

```
cfg <- list(n.sdt = 5, restriction = "standard-lure-pairs-eq", rec.pos =
c(1, 1, 6, 1, 6))
```

The configuration list provides the following information:

- Number of signals (`n.sdt = 5`)
- Types of restrictions (`restriction = "standard-lure-pairs-eq"`): The restriction specify a high threshold model for pairs with a lure distribution with equal standard deviations for signals of within a pair (cf. Chapter 3.5, for a detailed specification of the restrictions).
- Specification of the recollection positions (`rec.pos = c(1, 1, 6, 1, 6)`) for the five signals. Due to the fact that the recollection probability of the first distribution, representing the lures, is zero the first position is arbitrary.

4. The data are estimated, statistics are computed and the result is printed:

```
Opti1.Obj <- SDT.Estimate(data = data, Model.Id = "DPSDT", n = cfg, test =
T)
Stat1.Obj <- SDT.Statistics(Opti1.Obj)
print(Stat1.Obj)
```

The first command performs the estimation. The results of this process are stored in the object `Opti.Obj`.

The second command results in the computation of relevant statistics that are stored in the object `Stat.Obj`.

Finally, the content of `Stat.Obj` is printed.

The output provided by this piece of code looks like this:

```
------------------------------------------------------------------------
 DPSDT 1: Associative recognition (Data from Kelley and Wixted, Exp.2:
          restriction = "standard-lure-pairs-eq"
          - Distribution 1 = N(0,1) [Lure]
          - recollection[2] = 0, recollection[4] = 0
          - stddev[2] = stddev[3] = stddev[4] = stddev[5]
------------------------------------------------------------------------

=========================================================
 Symbolic and numeric gradients at the optimum:
 Length of difference vector  : 2.288471e-07
 Length of gradient at optimum: 4.431718e-05
=========================================================
                  Symbolic    Numeric Difference
t-1[1]          -2.141e-05 -2.142e-05      1e-08
t-2[1]           2.989e-05  2.991e-05     -2e-08
t-3[1]           8.600e-06  8.590e-06      1e-08
t-4[1]          -1.357e-05 -1.356e-05     -1e-08
t-5[1]           9.790e-06  9.800e-06     -1e-08
Mean[2]         -7.370e-06 -7.290e-06     -7e-08
Stddev[2]       -1.560e-06 -1.580e-06      2e-08
Recollection[3] -2.110e-06 -2.170e-06      6e-08
Mean[3]         -6.310e-06 -6.270e-06     -4e-08
Mean[4]         -1.037e-05 -1.057e-05      2e-07
Stddev[4]       -5.470e-06 -5.480e-06      1e-08
Recollection[5] -3.730e-06 -3.740e-06      2e-08
Mean[5]         -2.360e-06 -2.370e-06      0e+00
=========================================================
$Model.description
[1] "Dual Process SDT model: Number of Gaussian models: <5> Type of restrictions:
<standard-lure-pairs-eq> Recollection positions: <1, 1, 6, 1, 6>"

$Statistics
                                     Statistic
log L                                -5249.641
X^2                                     14.258
G^2                                     14.132
df                                      12.000
p(Y > X^2)                               0.285
```

```
p(Y > G^2)                                    0.292
AIC                                       10525.282
BIC                                       10606.574
CAICF                                     10701.966
ICOMP                                     10532.674
ICOMP.R                                   10506.932
Free Parameters                              13.000
Length of gradient at optimum                 0.000
Rank of Hessian                              13.000
Condition number of information matrix      3325.156
Rank of model matrix: t(J) * J               13.000
Condition number of model matrix           4427.466

$Free.parameters
                  Value SE(delta) SE(numerical) CFI-95(Lower) CFI-95(Upper)
t-1[1]           -0.105     0.029         0.029        -0.161        -0.049
t-2[1]            0.669     0.029         0.029         0.612         0.726
t-3[1]            1.395     0.039         0.039         1.319         1.471
t-4[1]            1.747     0.047         0.047         1.655         1.839
t-5[1]            2.227     0.064         0.064         2.102         2.353
Mean[2]           0.328     0.073         0.073         0.184         0.472
Stddev[2]         1.360     0.071         0.071         1.221         1.499
Recollection[3]   0.307     0.036         0.036         0.237         0.378
Mean[3]           0.863     0.125         0.125         0.617         1.109
Mean[4]           0.135     0.102         0.102        -0.066         0.336
Stddev[4]         1.847     0.116         0.116         1.619         2.075
Recollection[5]   0.296     0.156         0.156         0.000         0.601
Mean[5]           2.741     0.408         0.408         1.941         3.540

$Full.parametervector
                  DPSDT.1 DPSDT.2 DPSDT.3 DPSDT.4 DPSDT.5
Recollection-pos    1.000   1.000   6.000   1.000   6.000
Recollection        0.000   0.000   0.307   0.000   0.296
Mean                0.000   0.328   0.863   0.135   2.741
Stddev              1.000   1.360   1.360   1.847   1.847
t-1                -0.105  -0.105  -0.105  -0.105  -0.105
t-2                 0.669   0.669   0.669   0.669   0.669
t-3                 1.395   1.395   1.395   1.395   1.395
t-4                 1.747   1.747   1.747   1.747   1.747
t-5                 2.227   2.227   2.227   2.227   2.227
[...]
```

The table at the beginning of the output is presented because the flag `test=T` was set in the estimation procedure. By consequence, the numeric gradient at the optimum is computed using the function `grad()` from the R package `numDeriv` that was includes at the beginning of the code file.
As in the previous examples, the numeric and symbolic gradients are very close, and, in addition, the length of the gradient at the optimum is close to zero, indicating that a stationary point was found.

The model identification string indicates that the correct model was used and the statistics sections provides the relevant general statistical information (for details, cf. Chapter 2.2).

The next section lists the free parameters as well as estimated values and estimated standard errors.
The standard errors of parameters are estimated by the two methods described above (cf. Chapter 6.1 or 8.1) are identical indicating the absence of major problems.

The next section presents the full parameter vector. For each model the recollection probability, the mean and standard deviation as well as the decision bounds are shown. The first

row labeled `Recollection-pos` indicates the response categories selected in case of recollection.

9. Working Examples: High threshold model for rating data (HT.n)

The present chapter provides examples for fitting the double high-threshold model with probabilities (HT.n). to rating data from two types of experiments:

1. Data of Yonelinas (1994) with three types of items: Distractors, exclusion, and inclusion items;
2. Data of Kelley and Wixted (2001) on associative recognition.

9.1 Example 1: Fitting the HT.n model to exclusion-inclusion data

The present example demonstrates how to utilize the HT.n module for building a hybrid model consisting of a (unequal variance) signal detection model and a (double) high threshold model with more than one response option in case of recollection. By consequence, the resulting model is a generalization of the HTSDT.1 model.
The files:

```
HT-n.1-1 (Yonelinas, 1994, Exp.1).R
HT-n.1-2 (Yonelinas, 1994, Exp.2).R
HT-n.1-3 (Yonelinas, 1994, Exp.3).R
```

contain the R-code for fitting various HT.n model to the data from three experiments of Yonelinas (1994).
The structure of the model is depicted in Macho (2002, Figure 6, on p.30) and the fit statistics using the program Excel for fitting the model are presented in Macho (2002, Table 6, on p.35). The results reported by Macho (2002) are exactly identical to the actual results.
The present example shows an excerpt of the file for fitting data from Exp.2. For ease of exposition, the part of the code for loading the relevant source files is not shown.

```
# ============================================================
# Examples of the documentation of Model HT.n:
# HT.n-1.2: Exclusion vs. inclusion
# Data from Yonelinas (1994), Exp.2
#
# Date of creation: October, 2009
# Author: Siegfried Macho
# ============================================================
... [Code for loading relevant source files]

# DATA OF YONELINAS (1994), Experiment 2: SHORT LISTS
data.short <-   c(509, 495,  98,  30,  16,   4,   # Distractors
                  557, 263, 137,  78,  75,  42,   # Exclusion items
                   51, 109,  88, 105, 214, 585)   # Inclusion items

# DATA OF YONELINAS (1994), Experiment 2: LONG LISTS
data.long  <-   c(307, 530, 192,  79,  39,   5,  # Distractors
                  335, 308, 162, 178, 126,  43,  # Exclusion items
                   46, 179, 130, 189, 252, 356)  # Inclusion items
n <- 3

# Specification of constraints
# Distractors
fixed.dist <- matrix(c(rep(0, 6),  1,
                             1:6,  7), nr = 2, byrow = T)
```

```
# Fixed constraints exclusion / inclusion:
fixed.exc.inc <- matrix(c(rep(0, 6),        1,  1,
                          14:16, 20:22, 9, 18), nr = 2, byrow = T)

fixed <- cbind(fixed.dist, fixed.exc.inc)
ident <- matrix(c(11:13,   8, 10,
                  25:23,  17, 19), nr = 2, byrow = T)

# Compute start parameter vector
par <- SDT.HT.n.start.par(n, data.short)

# Add redundant parameters so that their positions can be seen in output
fixed.show <- HT.n.Redundant.to.Fixed(par, n, fixed, ident)

# Compute Parameter information
Par.Info <- SDT.Parameter.Info(par = par, n = n, Model.Id = "HT.n", fixed =
fixed.show, ident = ident)
cat("\n-----------------------------------------------------------------\n")
cat("HT.3: Yonelinas (1994), Exp.2, Parameter-Information:")
cat("\n-----------------------------------------------------------------\n")
print(Par.Info)

cat("\n-----------------------------------------------------------------\n")
cat(" HT.3: Yonelinas (1994), Exp.2, short lists")
cat("\n-----------------------------------------------------------------\n")
Opti1.Obj <- SDT.Estimate(data = data.short, n = n, Model.Id = "HT.n",
fixed = fixed, ident = ident, test = T)
Stat1.Obj <- SDT.Statistics(Opti1.Obj)
print(Stat1.Obj)
```

`... [Code for fitting additional models]`

The code comprises five main sections.

1. The relevant source files (`SDT-Main.R`, `SDT-Auxiliary.R`, and `SDT-HT-n.R`) are loaded. This section was omitted:

`... [Code for loading relevant source files]`

2. The two data sets (long vs. short lists) comprising three types of signals each (distracters, exclusion, and inclusion items) as well as the number of signals are specified:

```
data.short <-   c(509, 495,  98,  30,  16,   4,   # Distractors
                  557, 263, 137,  78,  75,  42,   # Exclusion items
                   51, 109,  88, 105, 214, 585)   # Inclusion items

# DATA OF YONELINAS (1994), Experiment 2: LONG LISTS
data.long  <-   c(307, 530, 192,  79,  39,   5,  # Distractors
                  335, 308, 162, 178, 126,  43,  # Exclusion items
                   46, 179, 130, 189, 252, 356)  # Inclusion items
n <- 3
```

3. Fixed and equality constraints are specified

```
fixed.dist <- matrix(c(rep(0, 6),  1,
                       1:6,  7), nr = 2, byrow = T)

# Fixed constraints exclusion / inclusion:
fixed.exc.inc <- matrix(c(rep(0, 6),        1,  1,
                          14:16, 20:22, 9, 18), nr = 2, byrow = T)

fixed <- cbind(fixed.dist, fixed.exc.inc)
ident <- matrix(c(11:13,   8, 10,
```

```
                    25:23,  17, 19), nr = 2, byrow = T)
```

The matrix `fixed.dist` specifies the constraints on the model for the distracter signal: The recollection probability parameter is set to zero and all rating probability parameters except for the last one is set to 0 (The last one is set to 1).
For the specification of further constraints, see the output below.

4. A table with parameter information is computed and printed. It enables one to check conveniently whether constraints on parameters are specified correctly. This is performed by the calling the function (cf. Chapter 2.4):

```
# Compute Parameter information
Par.Info <- SDT.Parameter.Info(par = par, n = n, Model.Id = "HT.n", fixed =
fixed.show, ident = ident)
print(Par.Info)
```

However, in order to show the redundant probability parameters (due to the fact that the rating parameters for each signal sum to 1.0) the parameter vector is generated and the redundant probability parameters are fixed:

```
# Compute start parameter vector
par <- SDT.HT.n.start.par(n, data.short)

# Add redundant parameters so that their positions can be seen in output
fixed.show <- HT.n.Redundant.to.Fixed(par, n, fixed, ident)
```

The function `HT.n.Redundant.to.Fixed` is called also during the setup of the model.

The table with the parameter information that is returned by the function `SDT.Parameter.Info()` looks like this:

```
$Model
[1] "HT.n: Double high threshold model for rating data"

$Parameters.and.Constraints
        name   par fixed.value ident.source Nr
1       p[1]  0.00           0          ---
2     p-1[1]  0.00           0          ---
3     p-2[1]  0.00           0          ---
4     p-3[1]  0.00           0          ---
5     p-4[1]  0.00           0          ---
6     p-5[1]  0.00           0          ---
7     p-6[1]  0.00           1          ---
8    mean[2]  0.00         ---          ---
9  stddev[2]  1.00           1          ---
10      p[2]  0.00         ---          ---
11    p-1[2]  0.00         ---          ---
12    p-2[2]  0.00         ---          ---
13    p-3[2]  0.00 Redundant-p          ---
14    p-4[2]  0.00           0          ---
15    p-5[2]  0.00           0          ---
16    p-6[2]  0.00           0          ---
17   mean[3]  0.00         ---      mean[2]  8
18 stddev[3]  1.00           1          ---
19      p[3]  0.00         ---         p[2] 10
20    p-1[3]  0.00           0          ---
21    p-2[3]  0.00           0          ---
22    p-3[3]  0.00           0          ---
23    p-4[3]  0.00         ---       p-3[2] 13
24    p-5[3]  0.00         ---       p-2[2] 12
```

```
25    p-6[3]  0.00         ---       p-1[2] 11
26       t-1 -1.50         ---          ---
27       t-2 -0.75         ---          ---
28       t-3  0.00         ---          ---
29       t-4  0.75         ---          ---
30       t-5  1.50         ---          ---
```

The table presents the name of the parameter in the first column, the actual value in the second column, the fixed constraints in the third column (numbers indicate the values and »---« indicate no constraints), the names of the source parameters of equality constraints in the fourth column, and the positions of the source parameters of equality constraints in the last column (cf. Chapter 2.4).

The entry `Redundant-p` for the third rating parameter for Signal 2 (representing the exclusion items) indicates that this probability parameter is redundant. By consequence, it is fixed internally by the program (using the function `HT.n.Redundant.to.Fixed`), in order to guarantee the identification of the model.

5. Finally, the model is fitted to the data of short lists and the results are evaluated:

```
Opti1.Obj <- SDT.Estimate(data = data.short, n = n, Model.Id = "HT.n",
fixed = fixed, ident = ident, test = T)
Stat1.Obj <- SDT.Statistics(Opti1.Obj)
print(Stat1.Obj)
```

The code for fitting additional models that is also contained in the file is not shown.

The output looks like this:

```
------------------------------------------------------------------------
 HT.3: Yonelinas (1994), Exp.2, short lists
------------------------------------------------------------------------
$Model.description
[1] "HT.n: Double high threshold model for rating data"

$Statistics
                                      Statistic
log L                                 -4578.589
X^2                                      28.279
G^2                                      26.538
df                                        6.000
p(Y > X^2)                                0.000
p(Y > G^2)                                0.000
AIC                                    9175.178
BIC                                    9230.509
CAICF                                  9300.544
ICOMP                                  9172.306
ICOMP.R                                9163.180
Free Parameters                           9.000
Length of gradient at optimum             0.000
Rank of Hessian                           9.000
Condition number of information matrix   333.278
Rank of model matrix: t(J) * J            9.000
Condition number of model matrix        655.198
$Free.parameters
         Value SE(raw parameters) SE(delta numeric) CFI-95(Lower) CFI-95(Upper)
mean[2]  1.374              0.062             0.062         1.253         1.496
p[2]     0.617              0.068             0.017         0.583         0.650
p-1[2]   0.754              0.245             0.030         0.694         0.813
p-2[2]   0.191              0.254             0.033         0.127         0.256
t-1     -0.106              0.036             0.036        -0.176        -0.036
t-2      1.063              0.043             0.043         0.979         1.146
t-3      1.603              0.050             0.050         1.505         1.701
t-4      2.039              0.061             0.061         1.919         2.159
```

```
t-5      2.704              0.082              0.082         2.543         2.865
```

... [Additional output, not shown]

The output is similar to those for the other models. There is however one difference: The standard errors of the recollection and rating probability parameters are computed by means of the delta method with the Jacobian of the transformation from raw to probability parameters being computed numerically. The result is presented in the third column [named `SE(delta numeric)`] in the section `$Free.parameters`.

9.2 Example 2: Fitting the HT.n model to associative recognition data

The files:

```
HT-n.2-1 (Associative Recognition, Kelley & Wixted, Exp.1).R
HT-n.2-2 (Associative Recognition, Kelley & Wixted, Exp.2).R
HT-n.2-3 (Associative Recognition, Kelley & Wixted, Exp.3).R
```

contain the R-code for fitting the HT.n model to the data from the three experiments of Kelley & Wixted (2001) on associative recognition. The following excerpt is contained in the file for fitting the data of Experiment 2. Again, for ease of exposition, only the relevant portions of the code are presented.

```
#=========================================================
# Examples of the documentation of Model HT.n:
# HT.n-2.2: Associative recognition
# Data from Kelley & Wixted (2001), Exp.2
# Date of creation: October, 2009
# Author: Siegfried Macho
#=========================================================
```

... [Code for loading relevant source files]

```
# DATA: Kelley & Wixted (2001, Exp.1 strong
# DATA OF KELLEY & WIXTED (2001), Experiment 1

data <- c(879, 553, 345,  61,  54,  28,   # Exp.2: Lure items
          179, 109,  82,  45,  28,  37,   # Exp.2: Weak   Items, rearranged
           81,  69,  62,  31,  37, 200,   # Exp.2: Weak   Items, intact
          215,  82,  60,  31,  31,  61,   # Exp.2: Strong Items, rearranged
           21,  24,  33,  21,  33, 348)   # Exp.2: Strong Items, intact

# NUMBER OF CONDITIONS (TYPES OF SIGNALS)
n <- 5

# Specification of constraints for HT.3
fixed.null <- matrix(c(rep(0, 6),  1,
                             1:6,  7), nr = 2, byrow = T)

fixed.weak <- matrix(c(rep(0, 6),
                           14:16, 20:22), nr = 2, byrow = T)

ident.weak <- matrix(c(11:13, 8:9,
                       25:23, 17:18), nr = 2, byrow = T)

fixed.strong <- matrix(c(rep(0, 6),
                           32:34, 38:40), nr = 2, byrow = T)

ident.strong <- matrix(c(11:13, 11:13, 26:27,
                           29:31, 43:41, 35:36), nr = 2, byrow = T)

fixed <- cbind(fixed.null, fixed.weak, fixed.strong)
```

```
ident <- cbind(ident.weak, ident.strong)

# Compute start parameter vector
par <- SDT.HT.n.start.par(n, data)

# Add redundant parameters so that ist position can be seen in output
fixed.show <- HT.n.Redundant.to.Fixed(par, n, fixed, ident)

# Compute Parameter information
Par.Info <- SDT.Parameter.Info(par = par, n = n, Model.Id = "HT.n", fixed =
fixed.show, ident = ident)
cat("\n-----------------------------------------------------------------\n")
cat("HT.3: Kelley & Wixted, Exp.2, Parameter-Information:")
cat("\n-----------------------------------------------------------------\n")
print(Par.Info)


cat("\n-----------------------------------------------------------------\n")
cat(" HT.3: Kelley & Wixted, Exp.2, Weak and strong items")
cat("\n-----------------------------------------------------------------\n")
Opti1.Obj <- SDT.Estimate(data = data, n = n, Model.Id = "HT.n", fixed =
fixed, ident = ident, test = T)
Stat1.Obj <- SDT.Statistics(Opti1.Obj)
print(Stat1.Obj)
```

`... [Code for fitting additional models]`

The code is quite similar to that of Example 1 (Section 9.1) for fitting the data of Yonelinas (1994). Again it comprises five main sections.

1. The relevant source files (`SDT-Main.R`, `SDT-Auxiliary.R`, and `SDT-HT-n.R`) are loaded. This section was omitted:

   `... [Code for loading relevant source files]`

2. The two data sets comprising five types of signals (Lure items, weak items – rearranged, weak items – intact, strong items – rearranged, and strong items – intact) as well as the number of signals are specified:

```
data <- c(879, 553, 345,  61,  54,  28,   # Exp.2: Lure items
          179, 109,  82,  45,  28,  37,   # Exp.2: Weak   Items, rearranged
           81,  69,  62,  31,  37, 200,   # Exp.2: Weak   Items, intact
          215,  82,  60,  31,  31,  61,   # Exp.2: Strong Items, rearranged
           21,  24,  33,  21,  33, 348)   # Exp.2: Strong Items, intact

# NUMBER OF CONDITIONS (TYPES OF SIGNALS)
n <- 5
```

3. Fixed and equality constraints are specified:

```
# Specification of constraints for HT.3
fixed.null <- matrix(c(rep(0, 6),  1,
                           1:6,  7), nr = 2, byrow = T)

fixed.weak <- matrix(c(rep(0, 6),
                       14:16, 20:22), nr = 2, byrow = T)

ident.weak <- matrix(c(11:13, 8:9,
                     25:23, 17:18), nr = 2, byrow = T)
```

```
fixed.strong <- matrix(c(rep(0, 6),
                         32:34, 38:40), nr = 2, byrow = T)

ident.strong <- matrix(c(11:13, 11:13, 26:27,
                         29:31, 43:41, 35:36), nr = 2, byrow = T)

fixed <- cbind(fixed.null, fixed.weak, fixed.strong)
ident <- cbind(ident.weak, ident.strong)
```

The matrix `fixed.null` specifies the constraints on the model for the lure signal: The recollection probability parameter is set to zero and all rating probability parameters except for the last one is set to 0 (The last one is set to 1).
For the specification of further constraints, see the output below.

4. A table with parameter information is computed and printed. It enables one to check conveniently whether constraints on parameters are specified correctly. This is performed by the call of the function (cf. Chapter 2.4):

```
# Compute Parameter information
Par.Info <- SDT.Parameter.Info(par = par, n = n, Model.Id = "HT.n", fixed =
fixed.show, ident = ident)
print(Par.Info)
```

However, in order to show the redundant probability parameters (due to the fact that the rating parameters for each signal sum to 1.0) the parameter vector is generated and the redundant probability parameters are fixed:

```
# Compute start parameter vector
par <- SDT.HT.n.start.par(n, data)

# Add redundant parameters so that ist position can be seen in output
fixed.show <- HT.n.Redundant.to.Fixed(par, n, fixed, ident)
```

The function `HT.n.Redundant.to.Fixed` is called also during the setup of the model.

The table with the parameter information that is returned by the function `SDT.Parameter.Info()` looks like this:

```
$Model
[1] "HT.n: Double high threshold model for rating data"


  $Parameters.and.Constraints
          name   par fixed.value ident.source Nr
  1       p[1]  0.00           0          ---
  2     p-1[1]  0.00           0          ---
  3     p-2[1]  0.00           0          ---
  4     p-3[1]  0.00           0          ---
  5     p-4[1]  0.00           0          ---
  6     p-5[1]  0.00           0          ---
  7     p-6[1]  0.00           1          ---
  8    mean[2]  0.00         ---          ---
  9  stddev[2]  1.00         ---          ---
  10      p[2]  0.00         ---          ---
  11    p-1[2]  0.00         ---          ---
  12    p-2[2]  0.00         ---          ---
  13    p-3[2]  0.00 Redundant-p          ---
  14    p-4[2]  0.00           0          ---
  15    p-5[2]  0.00           0          ---
  16    p-6[2]  0.00           0          ---
  17   mean[3]  0.00         ---      mean[2]  8
  18 stddev[3]  1.00         ---    stddev[2]  9
  19      p[3]  0.00         ---          ---
```

```
   20    p-1[3]  0.00           0          ---
   21    p-2[3]  0.00           0          ---
   22    p-3[3]  0.00           0          ---
   23    p-4[3]  0.00         ---       p-3[2] 13
   24    p-5[3]  0.00         ---       p-2[2] 12
   25    p-6[3]  0.00         ---       p-1[2] 11
   26   mean[4]  0.00         ---          ---
   27 stddev[4]  1.00         ---          ---
   28      p[4]  0.00         ---          ---
   29    p-1[4]  0.00         ---       p-1[2] 11
   30    p-2[4]  0.00         ---       p-2[2] 12
   31    p-3[4]  0.00         ---       p-3[2] 13
   32    p-4[4]  0.00           0          ---
   33    p-5[4]  0.00           0          ---
   34    p-6[4]  0.00           0          ---
   35   mean[5]  0.00         ---      mean[4] 26
   36 stddev[5]  1.00         ---    stddev[4] 27
   37      p[5]  0.00         ---          ---
   38    p-1[5]  0.00           0          ---
   39    p-2[5]  0.00           0          ---
   40    p-3[5]  0.00           0          ---
   41    p-4[5]  0.00         ---       p-3[2] 13
   42    p-5[5]  0.00         ---       p-2[2] 12
   43    p-6[5]  0.00         ---       p-1[2] 11
   44       t-1 -1.50         ---          ---
   45       t-2 -0.75         ---          ---
   46       t-3  0.00         ---          ---
   47       t-4  0.75         ---          ---
   48       t-5  1.50         ---          ---
```

The table presents the name of the parameter in the first column, the actual value in the second column, the fixed constraints in the third column (numbers indicate the values and »---« indicate no constraints), the names of the source parameters of equality constraints in the fourth column, and the positions of the source parameters of equality constraints in the last column.

The entry `Redundant-p` for the third rating parameter for Signal 2 (representing the weak rearranged items) indicates that this probability parameter is redundant. By consequence, it is fixed internally by the program (using the function `HT.n.Redundant.to.Fixed`), in order to guarantee the identification of the model.

5. Finally, the model is fitted to the data and the results are evaluated:

```
Opti1.Obj <- SDT.Estimate(data = data, n = n, Model.Id = "HT.n", fixed =
fixed, ident = ident, test = T)
Stat1.Obj <- SDT.Statistics(Opti1.Obj)
print(Stat1.Obj)
```

The code for fitting additional models that is also contained in the file is not shown.

The output resembles that of Example 1 (cf. Section 9.1).

10. Working Examples: Bivariate Gaussian model of signal detection (SDT.2D)

The present chapter illustrates the fitting of the bivariate Gaussian SDT model under different conditions. The first example demonstrates the fitting of the standard model without any additional complications. The second example demonstrates the treatment of structural zeros. The fourth example illustrates the fitting in case of violations of decisional separability. The fourth example demonstrates how to use a matrix for pooling data and estimates.

10.1 Example 1: Fitting the standard bivariate Gaussian model of signal detection

The present example demonstrates how to utilize the SDT.2D module for fitting bivariate signal detection data with no violations of decisional separability and no structural zeros.
The example file:

`SDT-2D-1-1. Estimation of Observer A of Thomas (2001).R`

contains the R code. The data of the present example stem from a face identification experiment of Thomas (2001, p.212, Table 6.1, Observer A). The data comprise responses to four types of faces resulting from varying two facial features with two values per feature. On each dimension, two response options were possible. This results in 4 (Faces [signals]) × 2 (Response options on Dimension 1) × 2 (Response options on Dimension 2) = 16 data points.
The structure of the fitted model (i.e., the location and orientation of the 90% confidence regions of the four Gaussian distributions as well as the location of the decision bounds) is shown in Figure 17.
For ease of exposition, in the following presentation the part of the R code for loading the relevant source files is not shown.

```
# ====================================================================
# SDT-2D-1-1:Example of the SDT-2D MODEL
#            Thomas, R. D. (2001). Characterizing perceptual interactions
#            in face identification using multidimensional signal detection
#            theory.
#            In M. J. Wenger, & J. T. Townsend (Eds.),
#            Computational geometry, and processing perspectives on facial
#            cognition
#            (Chapter 6, pp. 193-227). Hillsdale, NJ: Erlbaum.
#
# Data:      Observer A (p.212, Table, 6.1)
# Results:   p. 220, Table 7.
#
# Author:    Siegfried Macho
# Date of last update: September, 2013
# ====================================================================
```

`[...] [Code for loading relevant source files]`

```
# CONFIGURATION OF THE MODEL:
#--------------------------------------------------------------------
# Restrictions:   st = standard restrictions
#                 v1 = all variances equal to 1
#                 er = equal correlation coefficients
#                 qu = quadratic setup (rectangular configuration)
#--------------------------------------------------------------------
cfg <- list(n.sdt = 4, k1 = 1, k2 = 1, ds = 0, restriction = "st-v1-er-qu")

# DATA FROM THOMAS (2001)
data <- c(  83, 112,  47,  11,
            38, 154,  28,  33,
            15,  27, 117,  94,
             6,  36,  75, 136)

# PRINT PARAMETER INFORMATION
p.info <- SDT.Parameter.Info(n = cfg, Model.Id = "SDT.2D")
cat("\n=============================================")
cat("\n Parameter information:")
cat("\n=============================================\n")
print(p.info)
```

```
# CALL ESTIMATION MODULE AND PRINT THE RESULT
Opti.obj <- SDT.Estimate(data = data, n = cfg, Model.Id = "SDT.2D", test =
T)

# COMPUTE AND PRINT STATISTICS
Stat.Obj <- SDT.Statistics(Opti.obj)
cat("\n=============================================")
cat("\n Result of the Estimation: Thomas (2001):")
cat("\n=============================================\n")
print(Stat.Obj)

# PLOT CONFIGURATION
xy.labels <- c("Eyes", "Nose")
SDT.Plot(Opti.obj, cols = 3:6, labels = xy.labels)
```

This piece of code performs the following steps:

1. The list with the configuration of the model is specified:

```
cfg <- list(n.sdt = 4, k1 = 1, k2 = 1, ds = 0, restriction = "st-v1-er-qu")
```

The list contains the following entries:

`n.sdt` = Number of signals / Gaussian distributions;
`k1` = Number of decision bounds on Dimension 1;
`k2` = Number of decision bounds on Dimension 2;
`ds` = Flag indicating no violations of decisional separability;
`restriction` = String indicating the different types of restricitons (cf. Chapter 3.7):
`st` = standard restriction (for identification);
`v1` = all variance pararmeters equal to 1.0;
`er` = all correlation parameter are equal;
`qu` = rectangular setup.

Comment:
The entry in the list (`ds = 0`) is not required because the option of no violations is the default option.

2. The data vector is specified:

```
data <- c(  83, 112,  47,  11,
            38, 154,  28,  33,
            15,  27, 117,  94,
             6,  36,  75, 136)
```

Comment:
For information about the order of the data, cf. Chapter 3.7 and Figure 9.

3. The information concerning the starting parameter and the restrictions is computed and printed:

```
p.info <- SDT.Parameter.Info(n = cfg, Model.Id = "SDT.2D")
print(p.info)
```

Comment:
The content of the parameter vector is shown below in the output where the configuration of parameters and constraints is shown.

4. The estimation function is called:

```
Opti.obj <- SDT.Estimate(data = data, n = cfg, Model.Id = "SDT.2D" test =
T)
```

Comment:
No parameter vector is provided. By consequence, the vector of starting parameters is generated internally by the estimation function.

5. Test statistics are computed and printed:

```
Stat.Obj <- SDT.Statistics(Opti.obj)
print(Stat.Obj)
```

6. The configuration of Gaussian distributions is plotted:

```
xy.labels <- c("Eyes", "Nose")
SDT.Plot(Opti.obj, cols = 3:6, labels = xy.labels)
```

The plot function accepts as the first argument either an estimation object (generated by the function `SDT.Estimate`) or a list containing information about the model, the parameters, and the configuration (cf., Chapter 2.5, for details).

The R-code produces the following output:

```
================================================
 Parameter information:
================================================
$Model
[1] "Bivariate Gaussian SDT model: No violations of decisional separability
[RESTRICTIONS: st-v1-er-qu]"

$Parameters.and.Constraints
          name par fixed.value    ident.source Nr
1    Mean.1[1]   0     0 <set>             ---
2    Mean.2[1]   0     0 <set>             ---
3  Stddev.1[1]   1     1 <set>             ---
4  Stddev.2[1]   1     1 <set>             ---
5      Corr[1]   0         ---             ---
6    Mean.1[2]   0         --- Mean.1[1] <set>  1
7    Mean.2[2]   0         ---             ---
8  Stddev.1[2]   1     1 <set>             ---
9  Stddev.2[2]   1     1 <set>             ---
10     Corr[2]   0         ---   Corr[1] <set>  5
11   Mean.1[3]   0         ---             ---
12   Mean.2[3]   0         --- Mean.2[1] <set>  2
13 Stddev.1[3]   1     1 <set>             ---
14 Stddev.2[3]   1     1 <set>             ---
15     Corr[3]   0         ---   Corr[1] <set>  5
16   Mean.1[4]   0         --- Mean.1[3] <set> 11
17   Mean.2[4]   0         --- Mean.2[2] <set>  7
18 Stddev.1[4]   1     1 <set>             ---
19 Stddev.2[4]   1     1 <set>             ---
20     Corr[4]   0         ---   Corr[1] <set>  5
21      t.D1-1  -1         ---             ---
22      t.D2-1  -1         ---             ---

======================================================
 Symbolic and numeric gradients at the optimum:
 Length of difference vector  : 1.153319e-07
 Length of gradient at optimum: 9.986968e-07
======================================================
          Symbolic  Numeric Difference
Corr[1]   -6.4e-07 -6.4e-07    0.0e+00
Mean.2[2]  2.4e-07  2.4e-07    0.0e+00
Mean.1[3] -3.0e-07 -3.0e-07    0.0e+00
t.D1-1     6.5e-07  6.5e-07    0.0e+00
t.D2-1     1.1e-07  2.3e-07   -1.2e-07
======================================================

================================================
 Result of the Estimation: Thomas (2001):
```

```
================================================
$Model.description
[1] "Bivariate Gaussian SDT model: No violations of decisional separability
[RESTRICTIONS: st-v1-er-qu]"

$Statistics
                                        Statistic
log L                                   -1132.460
X^2                                         6.656
G^2                                         6.773
df                                          7.000
p(Y > X^2)                                  0.466
p(Y > G^2)                                  0.453
AIC                                      2274.919
BIC                                      2299.517
CAICF                                    2338.064
ICOMP                                    2266.641
ICOMP.R                                  2266.196
Free Parameters                             5.000
Length of gradient at optimum               0.000
Rank of Hessian                             5.000
Condition number of information matrix      9.108
Rank of model matrix: t(J) * J              5.000
Condition number of model matrix           14.392

$Free.parameters
           Value    SE CFI-95(Lower) CFI-95(Upper)
Corr[1]   -0.411 0.052        -0.512        -0.310
Mean.2[2]  0.604 0.079         0.449         0.759
Mean.1[3]  1.667 0.088         1.494         1.840
t.D1-1     0.714 0.061         0.595         0.832
t.D2-1     0.049 0.055        -0.058         0.157

$Gaussian.parameters
         Gauss-1 Gauss-2 Gauss-3 Gauss-4
Mean.1     0.000   0.000   1.667   1.667
Mean.2     0.000   0.604   0.000   0.604
Stddev.1   1.000   1.000   1.000   1.000
Stddev.2   1.000   1.000   1.000   1.000
Corr      -0.411  -0.411  -0.411  -0.411

$Decision.bounds.D1
              t-1
Dimension 1 0.714

$Decision.bounds.D2
              t-1
Dimension 2 0.049

$Data.and.Estimates
              Observed Expected Probabilities Residuals  Std.Resid.(analytical)
Data[1, 1, 1]       83   87.346         0.345    -0.465                  -0.809
Data[1, 1, 2]      112  105.509         0.417     0.632                   1.093
Data[1, 2, 1]       47   44.136         0.174     0.431                   0.614
Data[1, 2, 2]       11   16.008         0.063    -1.252                  -1.681
Data[2, 1, 1]       38   43.761         0.173    -0.871                  -1.316
Data[2, 1, 2]      154  149.094         0.589     0.402                   0.822
Data[2, 2, 1]       28   29.504         0.117    -0.277                  -0.370
Data[2, 2, 2]       33   30.640         0.121     0.426                   0.612
Data[3, 1, 1]       15   11.769         0.047     0.942                   1.222
Data[3, 1, 2]       27   31.294         0.124    -0.768                  -1.051
Data[3, 2, 1]      117  119.713         0.473    -0.248                  -0.454
Data[3, 2, 2]       94   90.224         0.357     0.398                   0.695
Data[4, 1, 1]        6    4.518         0.018     0.697                   0.825
Data[4, 1, 2]       36   38.545         0.152    -0.410                  -0.592
Data[4, 2, 1]       75   68.748         0.272     0.754                   1.190
```

```
Data[4, 2, 2]      136  141.190         0.558    -0.437                  -0.933
```

1. The table at the beginning of the output provides the relevant information about parameters and the restrictions imposed on them (cf., the discussion above). The marker `<set>` associated with restriction indicates that the restriction was set by the program.
2. The second table comparing the values of the numeric and symbolic gradient at the optimum was computed by the function `SDT.Estimate` since the flag `test = T` was set. Discrepancies between the numeric and analytic gradient indicate a failure of convergence. It is thus recommended to always use this option. In addition, the length of the symbolic gradient is close to zero indicating that the optimum was reached.

   Comment:
   In case of discrepancies between the numeric and analytic gradient set the flag `sym.gr = FALSE` and redo the estimation. In addition, contact me so that I can search for the reasons of the problem.
3. The output following to the table with symbolic and numeric gradients is provided by the function `SDT.Statistics` that computes the relevant information on the basis of the object provided by the estimation procedure (Details are given in Chapter 2.2).

   The numbers in the row labels `Data[s, D1, D2]` indicate the signal (`s`), the response category on Dimension 1 (`D1`) and the response category on Dimension 2 (`D2`) corresponding the actual data point, estimates, and residuals.
4. The graphical display of the configuration, shown in Figure 17, is generated by the function `SDT.Plot` that takes the information also from the object resulting from the estimation procedure (However, the function accepts also a different input format, cf. Chapter 2.5).

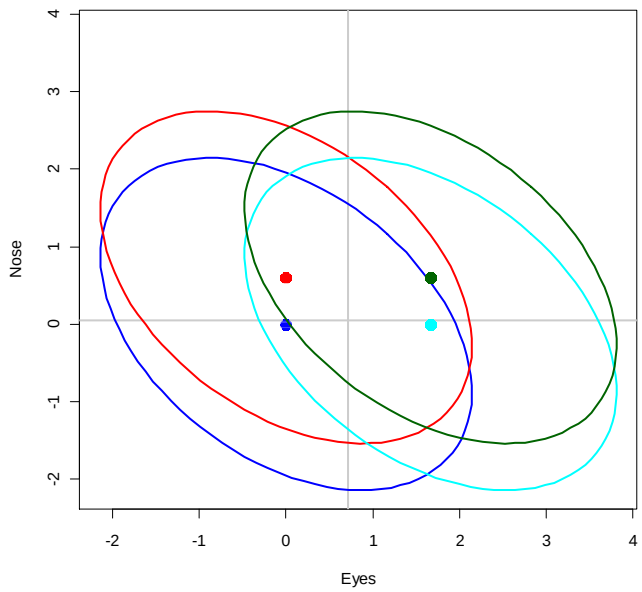


Figure 17: *SDT-2D model: Estimated Gaussian configuration and decision bounds for the data of Thomas (2001).*

10.2 Example 2: Fitting the bivariate Gaussian model of signal detection with structural zeros

The following example illustrates the specification of structural zeros. The corresponding file containing the example is named:

```
SDT-2D-2-1. Estimation of data from Olzak & Kramer (1984).R
```

The data of this example come from a study of Olzak and Kramer (1984). A detailed analysis of these data using the bivariate Gaussian SDT model was provided by Wickens and Olzak (1992).

The data comprise responses to four types of signals. On each dimension, six response options were possible. This results in 4 (Signals) × 6 (Response options on Dimension 1) × 6 (Response options on Dimension 6) = 144 data points.

The structure of the fitted model (i.e., the location and orientation of the 90% confidence regions of the four Gaussian distributions as well as the location of the decision bounds) is shown in Figure 18.

The R code for fitting these data differs in one significant respect from that of the previous example only. This concerns the list with the configuration information:

```
# CONFIGURATION OF THE MODEL
cfg <- list(n.sdt = 4, k1 = 5, k2 = 5, struct.zero = c(1, 36+1, 2*36+1,
3*36+1), restriction = "st-quvd")
```

The configuration list has been modified as follows:

1. The number of decision bounds `k1` and `k2` on Dimension 1 and 2, respectively has changed: Each dimension now comprises 5 decision bounds.
2. The entry `struct.zero` has been added. This entry contains the positions of structural zeros within the data vector. In the present case, the data on the positions 1, 37, 73, and 109 are specified as structural zeros.
3. The restriction have been changed:
   (i) `st` indicates standard restrictions, i.e. the menas and variance parameters of the first distribution are fixed.
   (ii) `quvd` indicates two types of restrictions: `qu` indicates a rectangular configuration and `vd` indicates dimensionwise equality of variances. The latter feature is illustrated in Figure 18: The variances of the first (dark blue) and second (cyan) Gaussian distribution are equal on the first dimension (low frequency signal). The variances of the first (dark blue) and third (red) Gaussian distribution are equal on the second dimension (high frequency signal). Finally, the variance of the fourth distribution (dark green) is equal to that of the second on the hight frequency dimension and to that of the third distribution on the low frequency dimension.

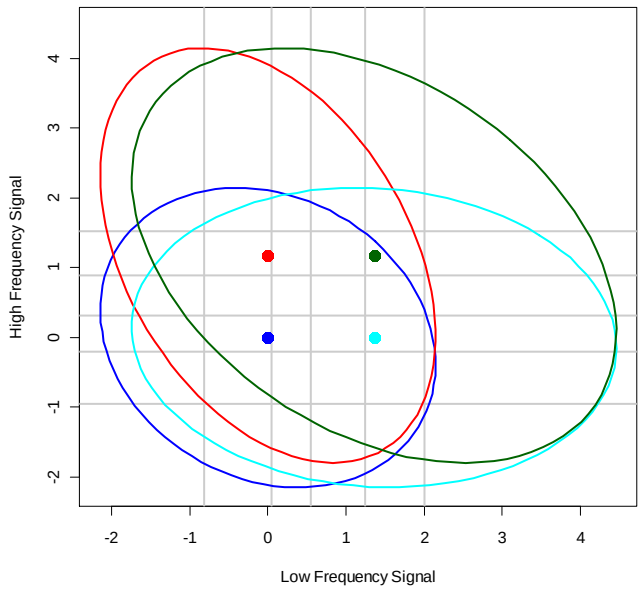


Figure 18: *SDT-2D model: Estimated Gaussian configuration and decision bounds for the data of Olzak & Kramer (1984) assuming decisional separability.*

Comment:
It is not required to set the frequencies in the data vector to zero. This is performed automatically by the program.

The following excerpt shows the relevant changes in the output due to the specification of structural zeros. The relevant rows with the data representing structural zeros are tagged by three stars `***`. In addition, the estimated probabilities, frequencies and residuals are all zero.

```
================================================
 Results for the data of Olzak and Kramer (1984):
================================================
$Model.description
[1] "Bivariate Gaussian SDT model: No violations of decisional separability
[RESTRICTIONS: st-quvd]"

$Statistics

[...]

$Data.and.Estimates
             Observed Expected Probabilities Residuals  Std.Resid.(analytical)
Data[1, 1, 1]     0***    0.000         0.000     0.000                   0.000
Data[1, 1, 2]     4      13.418         0.044    -2.571                  -3.025

[...]

Data[2, 1, 1]     0***    0.000         0.000     0.000                   0.000
Data[2, 1, 2]     4       2.691         0.008     0.798                   0.868

[...]

Data[3, 1, 1]     0***    0.000         0.000     0.000                   0.000
Data[3, 1, 2]     2       5.240         0.015    -1.416                  -1.561

[...]

Data[4, 1, 1]     0***    0.000         0.000     0.000                   0.000
Data[4, 1, 2]     1       0.583         0.002     0.547                   0.567

[...]
```

10.3 Example 3: Fitting the bivariate Gaussian model of signal detection with violations of decisional separability

The following example illustrates the fitting of a model that assumes violations of decisional separability on Dimension 1 using the same data as in the previous example. The name of the file with this example is:

```
SDT-2D-3-1. Estimation of data from Olzak & Kramer (1984) (ds = 1).R
```

This file differs only with respect to the content of the configuration list. Everything else is exactly the same (except from comments). The configuration list for this example is specified by the command:

```
# CONFIGURATION OF THE MODEL
cfg <- list(n.sdt = 4, k1 = 5, k2 = 5, ds = 1, restriction = "st-quvd")
```

Now the flag `ds = 1` was set to indicating violations of decisional separability on Dimension 1. In addition, no structural zeros were specified.

Comment:
The specification of structural zeros as in the previous example results in a rank deficient Jacobian matrix (More precisely: The column rank of the model matrix is not of full rank). Due to the fact that the warning flag is set the function `SDT.Statistics` prints the warning. The example file:

```
SDT-2D-6. Estimation of data from Olzak & Kramer (1984) (identification, ds
= 1, struct zero).R
```

contains the corresponding R code.

The following excerpt contains only those parts of the output that differ from the output of the previous examples:

```
================================================
 Results for the data of Olzak and Kramer (1984):
================================================
$Model.description
[1] "Bivariate Gaussian SDT model: Violation of decisional separability on
Dimension 1 [RESTRICTIONS: st-quvd]"

[...]

$Decision.bounds.D1
        D1: t-1 D1: t-2 D1: t-3 D1: t-4 D1: t-5
D2: t-1  -0.496  -0.137   0.236   0.873   1.382
D2: t-2  -1.506  -0.392   0.250   1.055   1.838
D2: t-3  -1.043  -0.163   0.453   1.137   1.989
D2: t-4  -1.183  -0.405   0.239   1.242   2.118
D2: t-5  -0.765   0.487   0.874   1.299   2.193
D2: t-6   0.275   0.954   1.276   1.769   2.257

$Decision.bounds.D2
               t-1    t-2   t-3   t-4   t-5
Dimension 2 -0.715 -0.067 0.424 0.982 1.594

[...]
```

1. The string with information on the model employed indicates that the SDT model assumeing violations of decisional separability was fitted.
2. The part of the output with the decision bounds now contains a table showing the decision bounds. Each row contains the decision bounds on Dimension 1 for a region on Dimension 2, for example the first row contains the decision bounds on Dimension 1 in within the region $\left(-\infty, t_1^{D2}\right]$ (see also Figure 19).

Figure 19 illustrates the configuration of Gaussian distributions as well as the locations of the decision bounds.

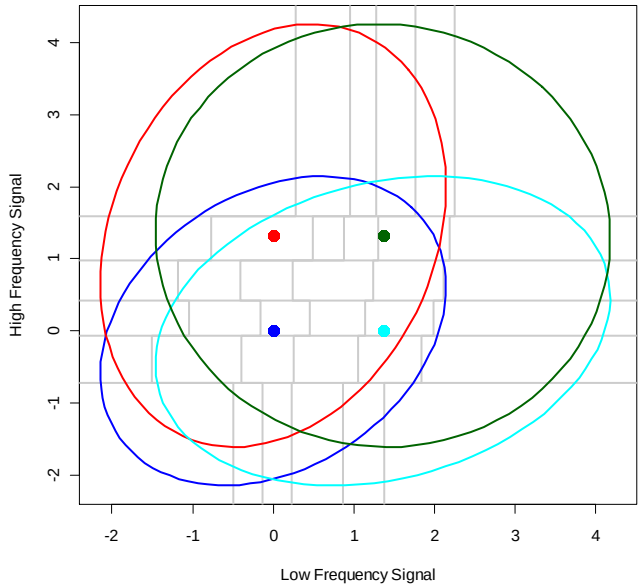


Figure 19: *SDT-2D model: Estimated Gaussian configuration and decision bounds for the data of Olzak & Kramer (1984) assuming violations of decisional separability on Dimension 1.*

Comment:
In order to fit the model assuming violations of decisional separability on Dimension 2 one has only to set `ds = 2` within the list with configuration information. The file with the code containing the example is named:

`SDT-2D-4-1. Estimation of data from Olzak & Kramer (1984) (ds = 2).R`

10.4 Example 4: Fitting the bivariate Gaussian model using a matrix for the pooling of data

In some applications it might be useful to pool some of the cells. This may be performed by specification of a matrix R (R = response pooling) in the configuration list. This matrix is used by the program to perform the transformation of the input data as well as the estimated probabilities and frequencies. Thus, in the course of estimation, the complete probability vector is computed (using the actual values of the parameters) and these probabilities are pooled by using matrix R. The resulting probabilities enter as probabilities into the (product) multinomial likelihood. By consequence, if the transformed and non-transformed counts follow a (product) multinomial distribution, the resulting estimates are maximum likelihood estimates. In case of pooling data corresponding to single signals this assumption is tenable. However, if matrix R is used to perform other types of transformations (e.g. differentially weighing the data points) the estimated parameters may not be maximum likelihood estimates.
The matrix R may be used to estimate bivariate Gaussian models with response selection (cf. Green, 2008, DeCarlo, 2003b).
The following example demonstrates the specification and usage of matrix R for pooling parts of the data. Again, data from Olzak & Kramer (1984) are used. The relevant R code is contained in the example file:

`SDT-2D-9-1. Estimation of data from Olzak & Kramer (1984) (Pooling of data).R`

The relevant code for specification of matrix R looks like this:

```
# SPECIFICATION OF A MATRIX FOR POOLING DATA
R <- diag(length(data))
group.1 <- c(2, 7, 8)        # Cells which are pooled with the target cells
group.2 <- c(2, 7, 8) + 36
group.3 <- c(2, 7, 8) + 36*2
group.4 <- c(2, 7, 8) + 36*3

target.1  <- 1                # Target cells
target.2  <- 1 + 36
target.3  <- 1 + 36*2
target.4  <- 1 + 36*3

# Matrix: Source cells in columns, target cells in rows
R[target.1, group.1] <- 1
R[target.2, group.2] <- 1
R[target.3, group.3] <- 1
R[target.4, group.4] <- 1

# Delete rows of cells that are grouped
R <- R[-c(group.1, group.2, group.3, group.4),]

# CONFIGURATION OF THE MODEL
cfg <- list(s = 4, k1 = 5, k2 = 5, ds = 0, R = R)
```

The code implements the following operations:

1. An identity matrix of order equal to the length of the data is generated by the command:
   ```
   R <- diag(length(data))
   ```
   In the present case the matrix is a 144 × 144 identity matrix (i.e. 1 on the main diagonal and 0 elsewhere).

2. The positions of the counts that are added to the target cells are specified:
   ```
   group.1 <- c(2, 7, 8)
   group.2 <- c(2, 7, 8) + 36
   group.3 <- c(2, 7, 8) + 36*2
   group.4 <- c(2, 7, 8) + 36*3
   ```
   In the present case the counts of the cells »surrounding« the first data points of each signal. are pooled to the counts of the first data points for each signal.

3. The positions of the four target cells are specified:
   ```
   target.1  <- 1
   target.2  <- 1 + 36
   target.3  <- 1 + 36*2
   target.4  <- 1 + 36*3
   ```
   In the present case, these are the first data points for each signal.

4. Ones are inserted into matrix R on the positions with rows corresponding to the target cells and columns to the source cells:
   ```
   R[target.1, group.1] <- 1
   R[target.2, group.2] <- 1
   R[target.3, group.3] <- 1
   R[target.4, group.4] <- 1
   ```
   Due to the fact that data vector and the vector of estimated probabilities are premultiplied by R this specification results in the addition of the counts in the target cells to those in the source cells.

5. Finally, the rows representing the source cells are deleted from the matrix:

```
R <- R[-c(group.1, group.2, group.3, group.4),]
```

Please note:

(1) The program assumes that the full data vector is provided as input. The pooling of data and estimates is performed internally by the program.

(2) Do not include structural zeros into the set of pooled data cells. The inclusion of structural zeros might result in a wrong computation of the degrees of freedom for to the χ^2 statistics.

As shown in the following excerpt the utilization of matrix R results in a slightly modified output of the data and estimate section:

```
[...]

$Data.and.Estimates
    Observed Expected Probabilities Residuals  Std.Resid.(analytical)
1         91   94.643         0.271    -0.374                  -2.464
2          9   16.474         0.047    -1.842                  -2.068
3          7   14.390         0.041    -1.948                  -2.210
4          6    8.608         0.025    -0.889                  -1.003

[...]
```

The modification concerns the numbering of rows on the left side. Due to the usage of matrix R it is not possible to specify for each data and estimate the signal and the number of response within each signal. By consequence, the rows are simply numbered.

10.5 Example 5: Robust estimation of decision bounds with pooled estimates

An application using a pooling matrix the estimation may fail if the decision bounds are not estimated using the `robust` option. The model discussed in the present chapter and its application to data provides a concrete instance.

The model depicted in Figure 20 was designed for modeling Remember-Know judgments (and conjoint recognition judgments, respectively). The model assumes two independent dimensions that are relevant for Remember-Know judgments: Familiarity (horizontal axis) and recollection (vertical axis). If the memory signal exceeds one of the decision bounds $\tau_1^R, \tau_2^R, \ldots \tau_5^R$ on the recollection dimension a »Remember« response is emitted, with confidence level determined by the decision bounds on the corresponding aixs (the vertical grey lines in Figure 20). If the meory strength is located below the lowest bound τ_1^R, the decision is guides by the degree of familiarity and the decision baounds $\tau_1^K, \tau_2^K, \ldots \tau_5^K$ on the familiarity axis are used for selecting a confidence category for the »Know« response. This type of model for explaining Remember-Know judgments has been proposed by Rotello, Macmillan, and Reeder (2004, Figure 12, on p. 602).

The model requires the pooling of cells: For example the region representing the »Remember« response with confidence level C_2 is made up of the cells 2, 8, 14, 20, 26, and 32. By consequence, these cells have to be pooled together. A similar pooling of cells has to be performed for the other »Remember« responses.

The estimation of the model to the data of Rotello, Mamillan, Reeder; & Wong (2005) without robust estimation of decision bounds fails to converge on a local optimum. Thus the robust option is required for fitting the model. The following file contains the relevant code for modeling these data:

`SDT-2D.12. (R-K Rotello, 2005, neutral condition, robust).R`

There are two specific requirements for fitting this model to the data using a pooling matrix:
1. The input data require a specific format, and
2. The pooling matrix has to be specified.

The structure of the input data is exhibited by following excerpt of code:

```
#                 Lure  Target
data <- matrix(c(745,  290,     # 1  New
                  46,   26,     # 2  Remember 2
                  29,   38,     # 3  Remember 3
                  47,   61,     # 4  Remember 4
                  51,  109,     # 5  Remember 5
                  55,  433,     # 6  Remember 6
                 199,  122,     # 7  Know 2
                   0,    0,     # 8  Remember 2
                   0,    0,     # 9  Remember 3
                   0,    0,     # 10 Remember 4
                   0,    0,     # 11 Remember 5
                   0,    0,     # 12 Remember 6
                 124,  106,     # 13 Know 3
                    [ ... ]
                  13,   59,     # 31 Know 6
                   0,    0,     # 32 Remember 2
                   0,    0,     # 33 Remember 3
                   0,    0,     # 34 Remember 4
                   0,    0,     # 35 Remember 5
                   0,    0      # 36 Remember 6
                   ), nc = 2, byrow = T)
```

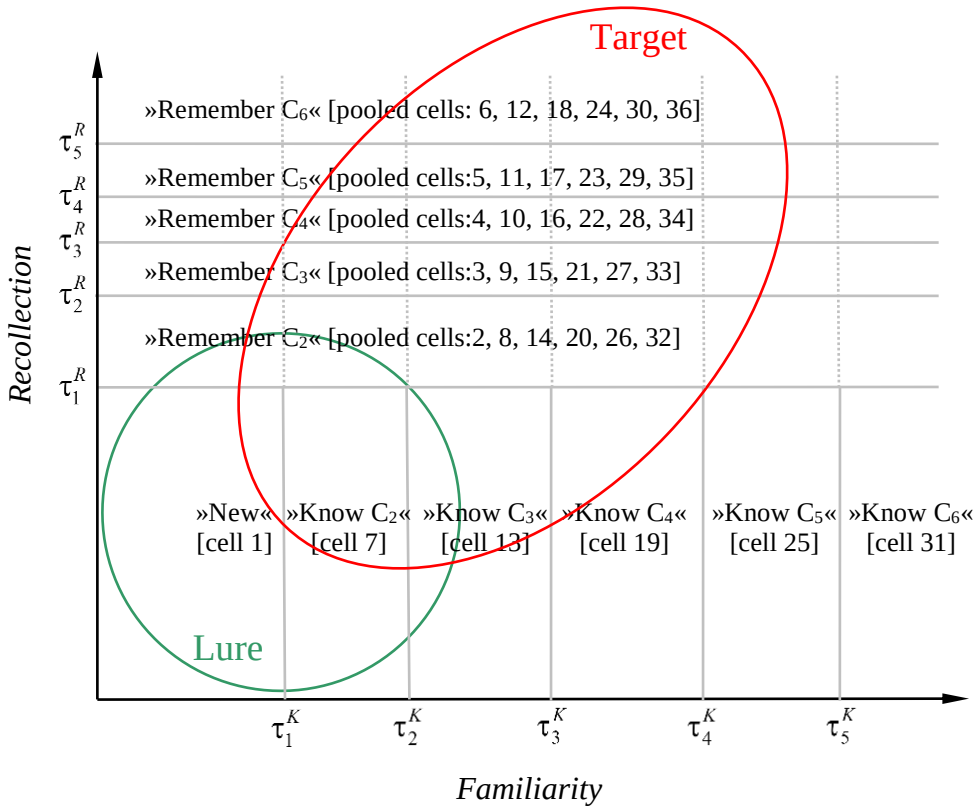


Figure 20: *Bivariate SDT model of Remember-Know recognition judgments: The two dimensions making up the decision space are recollection and familiarity.*

Horizonal grey lines indicate decision bounds on the recollection dimension and vertival lines indicate decision bounds on the familiarity dimension.

The first column represents the data for the lure (new) items for the different response categories and the second colum contains the respective data for old items.
The central feature of the data matrix consists in the fact that there are many rows of »pseudo data« with zero counts. These are required because the model requests 36 data points per signal type due to 5 decision bounds on each dimension. There are however only 13 response categories (1 »New«, 6 »Know« and 6 »Remember« responses). By consequence 23 »pseudo data« consisting of zeros have to be generated in order to fit the model. These counts are added by means of the pooling matrix to the main data, consistiong of the observed frequencies for the different response categories.
Consider, for example, the »Remember« response associated with with the lowest confidence (called Remember 2 in the code above). The corresponding response region consists of the cells: 2, 8, 14, 20, 26, 32 (cf. Figure 20). The counts of these cells (as well as the predicted frequencies) are pooled together. In the input, the observed frequencies of Remember 2 responses are associated with Cell 2 whereas the frequencies of the other cells are set to zero (Clearly, it makes no difference if the observed frequencies are associated with a different cell of the set or if they are distributed over the whole set, as long as the sum of the counts correspond to the observed values). The same principle was applied to the »Remember« responses associated with the other confidence categories.

The second important requirement for fitting the data concerns the specification of the pooling matrix R. Here is the relevant piece of R code:

```
R <- diag(length(data))

# Specify relevant entries for the lure signal
C2 <- c(8, 14, 20, 26, 32)
C3 <- C2 + 1
C4 <- C2 + 2
C5 <- C2 + 3
C6 <- C2 + 4

R[2, C2]   <- 1
R[3, C3]   <- 1
R[4, C4]   <- 1
R[5, C5]   <- 1
R[6, C6]   <- 1

# Specify relevant entries for the target signal
C22 <- C2 + 36
C23 <- C3 + 36
C24 <- C4 + 36
C25 <- C5 + 36
C26 <- C6 + 36
R[2+36, C22]   <- 1
R[3+36, C23]   <- 1
R[4+36, C24]   <- 1
R[5+36, C25]   <- 1
R[6+36, C26]   <- 1

# Delete rows of the cells that are grouped together
R <- R[-c(C2, C3, C4, C5, C6, C22, C23, C24, C25, C26),]
```

The first command:

```
R <- diag(length(data))
```

generates a 72×72 identity matrix (2 signals × 36 response regions). The command:

```
C2 <- c(8, 14, 20, 26, 32)
```

specifies the cells whose content is added to the second cell, representing Remember 2 responses for lure items. Similarly, the command:

```
C3 <- C2 + 1
```

specifies the cells whose content is added to the third cell, representing Remember 3 responses for lure items. `C3` results from `C2` by adding a 1 to each entry of `C2`.
The command:

```
R[2, C2]   <- 1
```

puts ones into columns 8, 14, 20, 26, and 32 (whose indices are represented by `C2`) of the second row of the pooling matrix `R`. As a result, if the data vector is multiplied from the right with matrix `R`, the entries 8, 14, 20, 26, and 32 are added to Entry 2. The following commands (shown above) perform the same procedure for the other rows (3-6). In this way, the pooling of data for lure items is ensured.
Now, the whole procedure is repeated for the old items. Finally, the command:

```
R <- R[-c(C2, C3, C4, C5, C6, C22, C23, C24, C25, C26),]
```

Deletes the rows corresponding to the cells that are added to Cells 2, 3, …, 42.

Following to the specification of the pooling matrix, the model setup is defined:

```
cfg <- list(n.sdt = 2, k1 = 5, k2 = 5, ds = 0, R = R, robust = T)
```

`n.sdt` denotes the number of signals, `k1` and `k2` refer to the number of response criteria on Dimension 1. and 2, `ds = 0` indicates decisional separabilty (this argument may by omitted because this option is the default one). Finally the pooling matrix `R` is passed and robust estimation of decision bounds is specified by setting `robust = T`.
If the robust estimation option is not set, the estimation does not converge properly as indicated by the following output:

```
=========================================================
 Symbolic and numeric gradients at the optimum:
 Length of difference vector  : 3.446338e+17
 Length of gradient at optimum: 3.446338e+17
=========================================================
                  Symbolic     Numeric    Difference
Mean.1[2]    1.661760e+02   166.17598 -1.600000e-07
Mean.2[2]   -9.490750e+00    -9.49075  2.400000e-07
Stddev.1[2]  5.504106e+01    55.04106 -6.000000e-08
Stddev.2[2] -9.320256e+01   -93.20256  0.000000e+00
t.D1-1      -1.207060e+02  -120.70603  3.000000e-08
t.D1-2      -7.509269e+02  -750.92690  8.000000e-08
t.D1-3      -1.385655e+02  -138.56550 -1.150000e-06
t.D1-4      -4.701655e+01   -47.01655 -3.000000e-08
t.D1-5      -9.855457e+01   -98.55457  5.000000e-08
t.D2-1       2.436929e+17 -1405.05202  2.436929e+17
t.D2-2      -2.436929e+17   140.11090 -2.436929e+17
t.D2-3       8.684503e+01    86.84503 -1.890000e-06
t.D2-4       2.133776e+02   213.37758 -1.600000e-07
t.D2-5       6.532763e+01    65.32763  7.000000e-08
=========================================================

===> WARNING: Hessian matrix is not positive definite: Rank = 13
```

On the one hand, there is a huge difference between the numeric and the analytic gradient computed at the estimated paramet values. This is due to the difference for first two decision

bounds on the second dimension. On the other hand, the Hessian matrix is singular (the rank of the matrix should be 14).
These difficulties arise because the code for computing probabilities makes some ad hoc adjustmenst in order to avoid negative probabilities. In the present case, these ad hoc adjustments are not sufficient to provide a proper results. With robust estimation negative probabilites due to a wrong ordering of decision bounds can never occur. By consequence, in case of robust estimation no problems are observed. These is indicated by the resulting table containing the computed gradients at the optimum (as well as by the absence of warning message concerning the Hessian matrix):

```
=======================================================
 Symbolic and numeric gradients at the optimum:
 Length of difference vector  : 1.343428e-07
 Length of gradient at optimum: 2.25306e-05
=======================================================
             Symbolic    Numeric Difference
Mean.1[2]   -9.84e-06 -9.890e-06      5e-08
Mean.2[2]    4.50e-07  4.300e-07      1e-08
Stddev.1[2] -5.44e-06 -5.430e-06      0e+00
Stddev.2[2]  1.94e-06  1.970e-06     -3e-08
t.D1-1       1.68e-05  1.674e-05      6e-08
t.D1-2       3.72e-06  3.750e-06     -3e-08
t.D1-3       5.82e-06  5.800e-06      2e-08
t.D1-4      -1.85e-06 -1.800e-06     -5e-08
t.D1-5       2.72e-06  2.660e-06      7e-08
t.D2-1      -4.67e-06 -4.620e-06     -5e-08
t.D2-2      -1.38e-06 -1.380e-06      0e+00
t.D2-3       4.00e-07  4.100e-07     -1e-08
t.D2-4       1.49e-06  1.470e-06      2e-08
t.D2-5      -3.23e-06 -3.210e-06     -2e-08
=======================================================
```

The table indicates proper convergence of the estimation procedure: The gradient at the estimated value is close to zero (about 0.00003), and the numerically and analytically computed gradients are nearly identical.

Comment:
Proper convergence also results without setting the robust option if the flag `use.nlminb = T` is set in the procedure `SDT.Estimate`. Since this is now the default option the robust flag needs no longer be set for this example.

Figure 21 depicts the 90 percent confidence regions and the location of the decision bounds. This picture was generated by using the commands:

```
xy.labels <- c("Familiarity", "Recollection")
SDT.Plot(Opti.obj, cols = 3:6, labels = xy.labels, option = 6:30)
```

The vector that is passed to argument `option` indicates the number of those decision bounds that are left out from the figure. It is assumed that the decision bounds on Dimension 1 are ordered as shown in Figure 7 and are located in front of decision bounds of Dimension 2 whose is ordering depicted in Figure 8. Figure 1 presents for a concrete example exhibiting the ordering of the decision bounds.

Comment:
All source files with the word »robust« included in the file name use the `robust = T` option. A comparison of the output with and without the option exhibits that the specification of the option has no effect on the output.

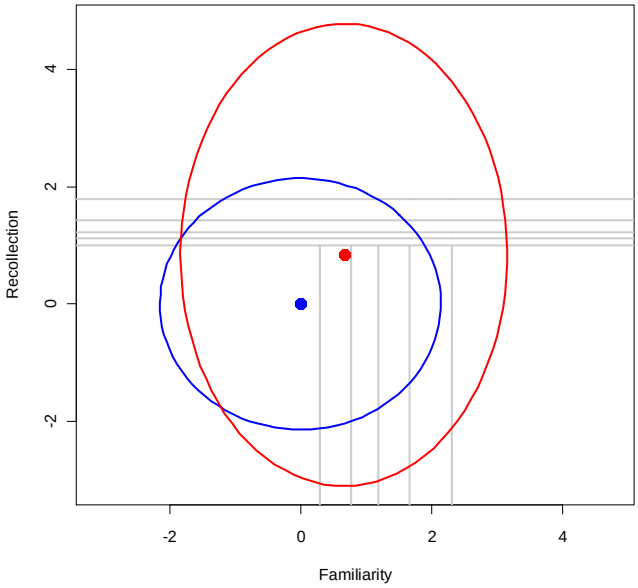


Figure 21: *Estimated configuration of the model for the actual data set.*

10.6 Example 6: Implementation of the CDP-SDT model

The CDP-SDT model (Continuous Dual-Process Signal Detection Model) of Wixted and Mickes (2010) can be implemented by means of the SDT-2D model. This implementation requires the specification of a response pooling matrix R as well as the definition of complex functional constraints.

In the following example the model was implemented to fit the data of a Remember/Know recognition experiment from Rotello, Macmillan, Reeder, & Wong (2005) [p. 873]. The following file contains the code of this example:

`SDT-2D-10-1. Implementation of the CDP-SDT Model.R`

The relevant code for specification of matrix R looks like this:

1. The data are specified:

```
# DATA from Rotello et al.(2005), NEUTRAL CONDITION
data <- matrix(c(
          # Lure  Target
            745,  290,  # Category 1 New
            0,    0,    # Category 1 New
            199,  122,  # Category 2 Know
            46,   26,   # Category 2 Remember
            124,  106,  # Category 3 Know
            29,   38,   # Category 3 Remember
            84,   101,  # Category 4 Know
            47,   61,   # Category 4 Remember
            45,   92,   # Category 5 Know
            51,   109,  # Category 5 Remember
            13,   59,   # Category 6 Know
            55,   433), # Category 6 Remember
            nc = 2, byrow = T)

dvec <- c(data[,1], data[,2])           # Make data vector
```

The data contain the Remember /Know [R/K] responses for lure (new) and target (old) items. Note that for Confidence Category 1 [sure new] no remember answers are present because in

case of selecting this response category no R/K responses were given. Thus, the data of R/K for this category are pooled together by using a pooling matrix R.

2. The pooling matrix R is specified as follows:

```
R <- diag(length(data))
R[1, 2]   <- 1
R[13, 14] <- 1
R <- R[-c(2, 14),]
```

The first two data cells (lure items) as well as 13. and 14. cell (target items) are pooled together and the redundant cells 2 and 14 are deleted. By consequence, instead of 24 data points, 22 are fitted only.

3. The configuration of the model is specified:

```
# CONFIGURATION OF THE MODEL
cfg <- list(n.sdt = 2, k1 = 5, k2 = 1, ds = 0, standard = F, R = R)
```

Note that the option `standard = F` was set indicating that the standard restrictions ($\mu_{11} = 0$, $\mu_{12} = 0$, $\sigma_{11} = 1$, and $\sigma_{12} = 1$) are not implemented automatically.
The reason is exhibited in Tab. 1 that shows the parameters together with the restrictions imposed:

Lure	$\mu_{11} = 0$	$\mu_{12} = 0$	$\sigma_{11} = \sqrt{2}$	$\sigma_{12} = 1$	$\rho_1 = 1/\sqrt{2}$
Target	μ_{21}	μ_{22}	$\sigma_{21} = \sqrt{1+\sigma_{22}^2}$	σ_{22}	$\rho_2 = \sigma_{22}/\sqrt{1+\sigma_{22}^2}$
Thresholds (D_1)	τ_{11}	τ_{12}	τ_{13}	τ_{14}	τ_{15}
Thresholds (D_2)	τ_{21}				

Tab. 1: *Parameter restrictions for implementing the CDP-SDT-Model for the data of Rotello et al. (2005)*

Instead of setting $\sigma_{11} = 1$ it is fixed at $\sigma_{11} = \sqrt{2}$. Note also that the implementation requires the definition of the functional constraints: $\sigma_{21} = \sqrt{1+\sigma_{22}^2}$ and $\rho_2 = \sigma_{22}/\sqrt{1+\sigma_{22}^2}$. This is performed by the following piece of code:

4. Specification of fixed and functional constraints is

```
# SPECIFICATION OF FIXED CONSTRAINTS:
fixed.par <-matrix(c(0, 0, sqrt(2), 1, 1/sqrt(2), 1,  1,
                     1, 2,       3, 4,         5, 8, 10), nrow = 2, byrow =
T)

#---------------------------------------------------------------------
# DEFINE FUNCTION FOR IMPLEMENTING FUNCTIONAL CONSTRAINTS
#---------------------------------------------------------------------
fct.res <- function(par)
{
      par[8]  <- sqrt(1 + par[9]*par[9])
      par[10] <- par[9]/par[8]

      # Gradient function
      gr.fct <- function(grad, par)
      {
            zw <- sqrt(1+ par[9]*par[9])
            grad[9]  <-  grad[9]  +  grad[8]*par[9]/zw  +  grad[10]*(1/zw  -
par[9]*par[9]/zw/zw/zw)
            return(grad)
```

```
      }
      attr(par, "SDT.gradient") <- gr.fct           # Gradient function
      return(par)
}
```

The function `fct.res` implements the functional constraints. In addition the gradient function `gr.fct` for adjusting the gradient is defined.

5. Specification of the function for computing the Jacobian matrix of the functional constraints:

```
#-------------------------------------------------------------------------
# DEFINE FUNCTION FOR COMPUTING THE JACOBIAN FOR THE ANALYTICAL COMPUTATION
# OF STANDARDIZED RESIDUALS
#-------------------------------------------------------------------------
J.fct <- function(par)
{
      zw <-  sqrt(1+ par[9]*par[9])
      J <- diag(length(par))
      J[8, 9]  <- par[9]/zw
      J[10, 9] <- 1/zw - par[9]*par[9]/zw/zw/zw
      J
}
attr(fct.res, "SDT.Jacobian") <- J.fct
```

This function is required for the computation of the analytic standardized residuals. Note that this function is not required because if it is not specified, the residuals are computed numerically (leading to the same result).

6. Now the estimation procedure may be called:

```
Opti.obj <- SDT.Estimate(data = dvec, n = cfg, Model.Id = "SDT.2D", fixed =
fixed.par, functional = fct.res, test = T)
```

7. Finally the statistics are computed and printed and the configuration is plotted:

```
Stat.Obj <- SDT.Statistics(Opti.obj)
print(Stat.Obj)

# PLOT CONFIRGUATION
xy.labels <- c("Confidence", "Recollection")
SDT.Plot(Opti.obj, cols = 3:6, labels = xy.labels, option = 11)
```

Figure 22 depicts the configuration. The blue ellipse represents lure items and the red one represents the target items. Note the the left part of the vertical decision bound was not plotted due to the command `option = 11` (cf. Chapter 2.5).
The resulting fit statistics of the model are identical to those reported by Wixted and Mickes (2010, p.1039).

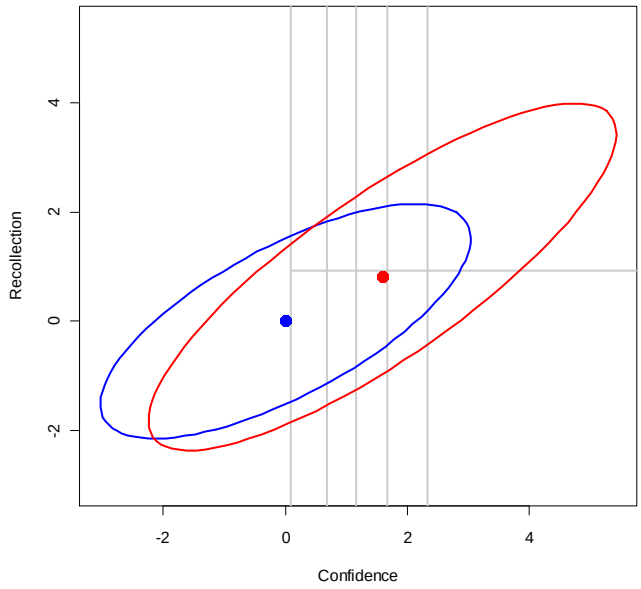


Figure 22: *SDT-2D model: Configuration of the CPD-SDT Model for the data of Rotello et al. (2005)*

The R file contains additional code for fitting data of the conservative condition of Rotello et al. (2005).

10.7 Source files containing further examples

The package comprises four additional source files (not mentioned above):

- `SDT-2D-6. Estimation of data from Olzak & Kramer (1984) (identification, ds = 1, struct zero).R`
- `SDT-2D-5. Estimation of data from Olzak & Kramer (1984) (functional constraints).R`
- `SDT-2D-5-1. Estimation of data from Olzak & Kramer (1984) (functional constraints, robust).R`
- `SDT-2D-8-1. Estimation of polychoric correlation (Finney & DiStefano).R`
- `SDT-2D-8-2. Estimation of polychoric correlation (Ollson).R`
- `SDT-2D-10-2. Implementation of the CDP-SDT Model (Violations of ds).R`
- `SDT-2D-11. Implementation of the CVM-SDT Model.R`

The first example demonstrates conditional estimation, i.e., in case of structural zeros, the residual probabilities of the signal are renormalized to sum to 1.0. The second and third file contain an example demonstrating the usage of functional constraints (without and with robust estimation). The fourth and fifth example illustrate the computation of the polychoric correlation coefficient. The sixth example implements the CDP-SDT model assuming violations of decisional separability for the conservative condition of Rotello et al. (2005). The final example implements the Criterion Variability Model for Remember-Know data of Rotello et al. (2005).

11. Working Example: Mixture of two Bivariate Gaussian models (SDT2D.MIX.2)

The present example demonstrates how to utilize the SDT2D.MIX.2 module for fitting bivariate signal detection data with violations of decisional separability on Dimension 1 and no structural zeros.

The example file:

`SDT2D-MIX-2 Sourcemonitoring Yonelinas (1999, Exp.2).R`

contains the R code. The data of the present example stem from a source monitoring experiment (Yonelinas, 1999, Experiment 2) consisting of new items and old items from two sources (male vs. female voice).

In the following, only the most important lines of the command file are presented. The lines containing the loading of source files and the data vectors are not shown. It should be noted however that the file SDT-SDT-2D.R containing the code of the SDT2D model must be included since the SDT2D.MIX.2 model uses functions of the SDT2D model.

Here is the code:

```
# CONFIGURATION INFORMATION
cfg <- list(s = 3, k1 = 5, k2 = 5, standard = F, ds = 1, robust = T)

#################################
# CONSTRAINTS
# D1: Source recognition
# D2: Old/New recognition
#
# Model 1: p, m1, m2, s1, s2, r:
#          1,  0,  0,  1,  1,  0,
#              0,  0,  1,  1,  0,
#
# Model 2: *,  0,  a,  1,  1,  0,
#              c,  b,  1,  1,  0,
#
# Model 3: *,  0,  a,  1,  1,  0,
#             -c,  b,  1,  1,  0,
#################################

# Fixed restrictions for Model 1
fixed.M1 <- matrix(c(1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0,
                     1:11), nr = 2, byrow = T)
# Fixed restrictions for Model 2
fixed.M2 <- matrix(c( 0,  1,  1,  0,  1,  1,  0,
                     13, 15, 16, 17, 20, 21, 22), nr = 2, byrow = T)

# Fixed restrictions for Model 3
fixed.M3 <- matrix(c( 0,  1,  1,  0,  1,  1,  0,
                     24, 26, 27, 28, 31, 32, 33), nr = 2, byrow = T)

# Target of functional constraints
fixed.fct <- matrix(c( 0,
                       29), nr = 2, byrow = T)

fixed <- cbind(fixed.M1, fixed.M2, fixed.M3, fixed.fct)
ident <- matrix(c(14, 19,
                  25, 30), nr = 2, byrow = T)

# Functional constraint:
# Mean.x of Model 2, second Gaussian = Mean.x of Model 3, second Gaussian
fct.m <- function(par)
{
      tindex <- 29                  # Target index: Mean.x[3.2]
      sindex <- 18                  # Source index: Mean.x[2.2]

      par[tindex] <- -par[sindex]

      # Define Gradient function
      gr.fct <- function(grad, par)
```

```
      {
            grad[sindex] <- grad[sindex] - grad[tindex]
            return(grad)
      }
      attr(par, "SDT.gradient") <- gr.fct             # Gradient

      return(par)
}
# 4 DEFINE FUNCTION TO COMPUTE Jacobian AND ASSIGN IT TO THE FUNCTION
J.fct <- function(par)
{
      J <- diag(length(par))
      J[29, 18] <- -1
      J
}
attr(fct.m, "SDT.Jacobian") <- J.fct                  # Jacobian matrix

#-------------------------------------------------------------------------
# SHOW THE CONFIGURATION
#-------------------------------------------------------------------------
Par.Info <- SDT.Parameter.Info(n = cfg, Model.Id = "SDT2D.MIX.2", fixed =
fixed, ident = ident, deci = 3)
print(Par.Info)

#-------------------------------------------------------------------------
# ESTIMATION OF PARAMETERS AND EVALUATION OF RESULTS
#-------------------------------------------------------------------------
Est.Obj <- SDT.Estimate(data = data.vec, Model.Id = "SDT2D.MIX.2", n = cfg,
fixed = fixed, ident = ident, functional = fct.m, test = T)
Stat.Obj <- SDT.Statistics(Est.Obj)

cat("\n-----------------------------------------------------------------\n")
cat(" Yonelinas (1999), Experiment 2:")
cat("\n-----------------------------------------------------------------\n")
print(Stat.Obj)

#-------------------------------------------------------------------------
# PLOT THE CONFIGURATION:
# Gaussian of New in blue, the other Gaussians are red
#-------------------------------------------------------------------------
SDT.Plot(Est.Obj, cols = c(1, 2, 2, 2, 2), labels = c("Source", "Old/New"))
```

The first command line:

```
cfg <- list(s = 3, k1 = 5, k2 = 5, standard = F, ds = 1, robust = T)
```

speicifies the configuration of the model:

- Three mixures of Gaussians are used: `s = 3`
- The number of thresholds on Dimension 1 (source dimension) and on Dimension 2 (Old/new recognition dimension) are both 5: `k1 = 5, k2 = 5`
- The standard option for fixing parameters is not used: `standard = F`
  Two things should be noted however: First, the actual fixation of parameters corresponds to the standard (cf. the entries in the matrix `fixed`), and, second, it does not matter whether `standard = F` or `standard = T` since the program eliminates redundant fixed constraints.
- Violation of decisional separability on Dimension 1 is assumed: `ds = 1`
- Robust estimation of decision bounds is performed: `robust = T`

The following comments indicate which parameters are fixed or set equal: '`*`' indicate free parameters, letters `a`, `b` and `c` indicate equality constraints (note the utilization of letters `c` and `-c` indicate that the second value was set to minus the first one.
The two Gaussians of the first model have the same parameters (all fixed) thus »simulating« a single Gaussian distribution. Due to the fact, the the probability parameter was set to 1.0, the second Gaussian distribution of the model has no impact on the result. By consequence, fhe parameters of this Gaussian may thus be fixed to any (permissible) set of values without changing the result. It is however important to fix these parameters since otherwise the model would not be identified.

The matrices `fixed.M1, fixed.M2 and fixed.M3` contain fixed constraints for each of the three models. The matrix `fixed.fct` represent the functional constraint. These matrices of constraints are binded together resulting in the matrix `fixed`.

The matrix `ident` represents the equality constraints. In the present case, the means on the *y*-dimension of Model 2 were equated to the corresponding means of Model 3.

The following lines are required for specifying the functional constraint that the mean on the *x*-dimension of the second Gaussian of Model 3 is set to minus the respective mean of Model 2 (cf. Figure 23, the two red circles in the upper part of the figure). The code is practically identical to the one described in Section 7.2. Thus, I refer readers to the description in this section (for further explications and examples concnering functional constraints, see Section 4.4.2).

The commands:

```
Par.Info <- SDT.Parameter.Info(n = cfg, Model.Id = "SDT2D.MIX.2", fixed =
fixed, ident = ident, deci = 3)
print(Par.Info)
```

display the parameters as well as the constraints specified. This can be very helpful for checking whether constraints were specified properly.

The estimation of the model is performed by the command:

```
Est.Obj <- SDT.Estimate(data = data.vec, Model.Id = "SDT2D.MIX.2", n = cfg,
fixed = fixed, ident = ident, functional = fct.m, test = T)
```

with the result being saved in the object `Est.Obj`.

The relevant statistical information is computed by the command:

```
Stat.Obj <- SDT.Statistics(Est.Obj)
```

Printing the contents of the object `Stat.Obj` results in the following output (outlined):

```
$Model.description
[1] "Bivariate Gaussian 2 Mixture model: Violation of decisional separability on
Dimension 1 [ROBUST]"

$Statistics
                                        Statistic
log L                                -16902.466
X^2                                     90.785
G^2                                     93.027
df                                      65.000
p(Y > X^2)                               0.019
p(Y > G^2)                               0.013
AIC                                  33884.933
BIC                                  34151.280
CAICF                                34454.119
ICOMP                                34002.844
ICOMP.R                              33839.370
Free Parameters                         40.000
```

```
Length of gradient at optimum                0.000
Rank of Hessian                             40.000
Condition number of information matrix 1453621.124
Rank of model matrix: t(J) * J              40.000
Condition number of model matrix         15578.430

$Free.parameters
                Value    SE CFI-95(Lower) CFI-95(Upper)
p[2]            0.644 0.025         0.595         0.692
Mean.y[2.1]     0.548 0.051         0.449         0.647
Mean.x[2.2]     0.995 0.208         0.588         1.402
Mean.y[2.2]     3.048 3.805        -4.411        10.506
p[3]            0.616 0.041         0.536         0.696
t.D1-1[D2 = 1] -1.979 0.014        -2.006        -1.952
t.D1-2[D2 = 1] -1.138 0.081        -1.297        -0.978
t.D1-3[D2 = 1]  0.411 0.041         0.330         0.492
t.D1-4[D2 = 1]  1.707 0.085         1.541         1.874
t.D1-5[D2 = 1]  2.146 0.118         1.913         2.378
t.D1-1[D2 = 2] -1.954 0.119        -2.187        -1.720
t.D1-2[D2 = 2] -1.197 0.136        -1.464        -0.931
t.D1-3[D2 = 2]  0.371 0.036         0.301         0.442
t.D1-4[D2 = 2]  1.921 0.085         1.754         2.087
t.D1-5[D2 = 2]  2.466 0.138         2.195         2.737
t.D1-1[D2 = 3] -2.187 0.139        -2.460        -1.915
t.D1-2[D2 = 3] -1.078 0.171        -1.412        -0.744
t.D1-3[D2 = 3]  0.294 0.043         0.211         0.378
t.D1-4[D2 = 3]  1.843 0.091         1.664         2.022
t.D1-5[D2 = 3]  2.590 0.176         2.245         2.936
t.D1-1[D2 = 4] -1.813 0.177        -2.159        -1.466
t.D1-2[D2 = 4] -0.877 0.192        -1.252        -0.501
t.D1-3[D2 = 4]  0.310 0.050         0.212         0.407
t.D1-4[D2 = 4]  1.210 0.067         1.079         1.341
t.D1-5[D2 = 4]  2.416 0.126         2.169         2.662
t.D1-1[D2 = 5] -1.567 0.077        -1.717        -1.416
t.D1-2[D2 = 5] -0.301 0.059        -0.417        -0.184
t.D1-3[D2 = 5]  0.172 0.081         0.013         0.331
t.D1-4[D2 = 5]  0.652 0.084         0.487         0.816
t.D1-5[D2 = 5]  1.812 0.074         1.668         1.956
t.D1-1[D2 = 6] -0.623 0.042        -0.706        -0.541
t.D1-2[D2 = 6] -0.201 0.071        -0.339        -0.062
t.D1-3[D2 = 6]  0.102 0.082        -0.059         0.263
t.D1-4[D2 = 6]  0.281 0.106         0.073         0.490
t.D1-5[D2 = 6]  0.702 0.071         0.562         0.842
t.D2-1         -0.671 0.029        -0.728        -0.614
t.D2-2          0.063 0.029         0.007         0.119
t.D2-3          0.552 0.034         0.484         0.619
t.D2-4          0.992 0.042         0.910         1.074
t.D2-5          1.616 0.054         1.511         1.721

$Gaussian.parameters
           Mixture-1 Mixture-2 Mixture-3
p                  1     0.644     0.616
Mean.x[1]          0     0.000     0.000
Mean.y[1]          0     0.548     0.548
Stddev.x[1]        1     1.000     1.000
Stddev.y[1]        1     1.000     1.000
Corr[1]            0     0.000     0.000
Mean.x[2]          0     0.995    -0.995
Mean.y[2]          0     3.048     3.048
Stddev.x[2]        1     1.000     1.000
Stddev.y[2]        1     1.000     1.000
Corr[2]            0     0.000     0.000

$Decision.bounds.D1
        D1: t-1 D1: t-2 D1: t-3 D1: t-4 D1: t-5
D2: t-1  -1.979  -1.138   0.411   1.707   2.146
```

```
D2: t-2  -1.954  -1.197   0.371   1.921   2.466
D2: t-3  -2.187  -1.078   0.294   1.843   2.590
D2: t-4  -1.813  -0.877   0.310   1.210   2.416
D2: t-5  -1.567  -0.301   0.172   0.652   1.812
D2: t-6  -0.623  -0.201   0.102   0.281   0.702

$Decision.bounds.D2
               t-1    t-2   t-3   t-4   t-5
Dimension 2 -1.954 -1.197 0.371 1.921 2.466

$Data.and.Estimates
             Observed Expected Probabilities Residuals  Std.Resid.(analytical)
Data[1, 1, 1]        9   11.529         0.006    -0.745                  -1.254
Data[1, 1, 2]       14   13.353         0.007     0.177                   0.256

[...]
```

Concerning the listing of results two things should be mentioned:

1. The model descriptions at the beginning of the listing provides information about the model and its configuration.
2. The condition number of the information matrix is quite high, indicating that the model might be too complex for the data set used. This is presumably due to the fact that a number of cells have low frequencies.

The command

```
SDT.Plot(Est.Obj, cols = c(1, 2, 2, 2, 2), labels = c("Source", "Old/New"))
```

Generates the plot of Figure 23.

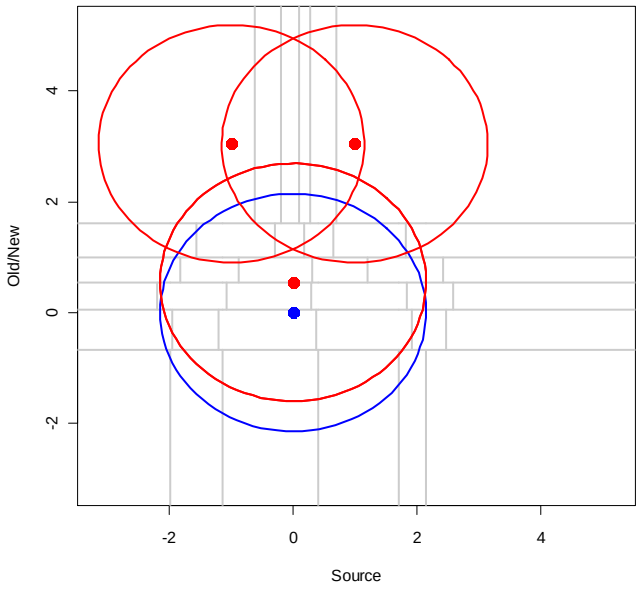


Figure 23: *SDT2D-MIX-2 model: Configuration for modeling the data of Yonelinas (1999, Experiment 2).*

12. Working Example: SDT model for m-alternative forced choice with bias (mAFC)

The present example demonstrates the utilization of the SDT.mAFC module for fitting 3-AFC forced choice data, assuming bias. The data come from a taste experiment performed by Ennis & O'Mahony (1995, Table 1 on p.1089).

Here is an excerpt from the script that performs the estimation and computation of test statistics for the WW condition. The file `SDT-mAFC (Ennis & Mahoney, 1995).R` contains the code.

```
# Data of Ennis & Mahony (1995), p. 1089, Condition WW
datavec1 <- c(54,  5,  1,        # SWW
              0, 60,  0,         # WSW
              5,  3, 52)         # WWS

# Data of Ennis & Mahony (1995), p. 1089, Condition SS
datavec2 <- c(40, 12,  8,        # SWW
              6, 49,  4,         # WSW
              6,  5, 49)         # WWS

# CONFIGURATION
cfg <- list(s = 1, m = 3, quad = "aGH")

# SET FIXED CONSTRAINT: FIX VARIANCE OF TARGET TO 1.0
fixed <- matrix(c(1, 2), nr = 2, byrow = T)

# ESTIMATE AND EVALUATE THE RESULTS
Opti.Obj <- SDT.Estimate(data = datavec1, n = cfg, Model.Id = "mAFC", fixed
= fixed, test = T)
Stat.Obj <- SDT.Statistics(Opti.Obj, deci = 3)

cat("\n-----------------------------------------------------\n")
cat(" 1. Ennis & Mahony (1995), Condition WW\n")
cat("    using adaptive Gauss-Hermite Quadrature")
cat("\n-----------------------------------------------------\n")
print(Stat.Obj)
```

The following steps are performed:

1. The two data vectors `datavec1` and `datavec2` from two experimental conditions WW and SS are specified.
2. The configuration is specified: `cfg <- list(s = 1, m = 3, quad = "aGH")`
3. The variance parameter (parameter in the second position [cf. Chapter 3.9]) is fixed to 1.0: `fixed <- matrix(c(1, 2), nr = 2, byrow = T)`
4. The estimation is performed via the `SDT.Estimate()` command with the model identification `Model.Id = "mAFC"`, and the result is assigned to the object `Opti.Obj`.
5. Test statistics are computed by means of the function `SDT.Statistics()`, and result is assigned to the object `Stat.Obj`.
6. The content of the Object `Stat.Obj` is printed: `print(Stat.Obj)`.
7. The output looks like this:

```
=====================================================
 Symbolic and numeric gradients at the optimum:
 Length of difference vector  : 2.263725e-09
 Length of gradient at optimum: 4.169881e-12
=====================================================
   Symbolic Numeric Difference
d'        0       0          0
b1        0       0          0
b2        0       0          0
=====================================================

-----------------------------------------------------
 1. Ennis & Mahony (1995), Condition WW
    using adaptive Gauss-Hermite Quadrature
-----------------------------------------------------
$Model.description
[1] "3-AFC Model with Bias, Assuming Gaussian Distributions, Number of Models: 1,
Method of quadrature: Adaptive Gauss-Hermite, Number of quadrature points: 25"
```

```
$Statistics
                                        Statistic
log L                                     -54.177
X^2                                         5.213
G^2                                         6.230
df                                          3.000
p(Y > X^2)                                  0.157
p(Y > G^2)                                  0.101
AIC                                       114.353
BIC                                       123.932
CAICF                                     138.592
ICOMP                                     109.285
ICOMP.R                                   109.198
Free Parameters                             3.000
Length of gradient at optimum               0.000
Rank of Hessian                             3.000
Condition number of information matrix       6.599
Rank of model matrix: t(J) * J              3.000
Condition number of model matrix           20.392

$Free.parameters
   Value    SE CFI-95(Lower) CFI-95(Upper)
d' 2.628 0.230         2.177         3.079
b1 0.400 0.282        -0.153         0.953
b2 0.854 0.309         0.248         1.460

$Full.parametervector
   Parameters
d'      2.628
sd      1.000
b1      0.400
b2      0.854

$Data.and.Estimates
           Observed Expected Probabilities Residuals  Std.Resid.(analytical)
Data[1, 1]       54   55.663         0.928    -0.223                  -1.844
Data[1, 2]        5    3.593         0.060     0.742                   1.199
Data[1, 3]        1    0.744         0.012     0.297                   0.391
Data[1, 4]        0    0.845         0.014    -0.919                  -1.213
Data[1, 5]       60   58.783         0.980     0.159                   1.846
Data[1, 6]        0    0.372         0.006    -0.610                  -0.736
Data[1, 7]        5    2.689         0.045     1.409                   2.203
Data[1, 8]        3    5.720         0.095    -1.137                  -1.977
Data[1, 9]       52   51.591         0.860     0.057                   0.445
```

1. The first table is printed by the estimation procedure `SDT.Estimate()` since the flag `test = T` was set. It shows that the estimation procedure converged on a local optimum with the numerically and analytically computed gradients being practically identical.
2. The second part of the ouput shows the content of the object `Stat.Obj` that was computed by the function `SDT.Statistics()`.
   The output comprises the following sections:
   (i) `Model.description:` A verbal description of the model;
   (ii) `Statistics:` A list of various test statistics;
   (iii) `Free.parameters:` Estimated parameters with estimated standard errors and confidence intervals;
   (iv) `Full.parametervector:` A list of all model parameters including constrained parameters;
   (v) `Data.and.Estimates:` Observed and estimated frequencies and probabilities as well as residuals;

Comment: Further details concerning the output of `SDT.Statistics()` are provided in Chapter 2.2.

The estimated parametes differ slightly from those reported by DeCarlo (2012, Table 3 on p.203). This is due to the fact that the present estimation procedure differs slightly from that used by DeCarlo. A comparison of both methods revealed that the present method of estimation is slightly superior with respect to precision of parameter estimation, coverage probabilities of estimated confidence intervals, as well as the power to detect whether parameters are different from zero.[1]

13. Working Example: SDT model for modeling forced choice and rating data (SDT.Rank)

In the following, two examples of the usage of the module SDT.Rank are presented. The first illustrates the modeling of pure AFC data with repeated choices (or ranking). The second example illustrates the joint modeling of ratning and AFC data.

13.1.1 SDT-Rank Example 1: Modeling pure forced choice with repeated choices

The present example demonstrates the utilization of the `SDT.Rank` module for fitting forced choice data or ranking data only. The data come from an own experiment. The first 16 data points result from a condition where participant had to rank the four items (one target and three distractors) presented [= parallel condition]. The second 16 data points come from a condition in which, in case of failing to choose the target item, the item set is reduced and participants had to choose from the set of remaining [= serial condition]. The 16 data points represent the number of cases where the target is chosen (ranked) as the first, second, third or last item, when the target was on the first second, third, and fourth position within the (full) distractor set.

Here is an extract of R code for modeling the two sets 4-AFC data

```
# DATA FROM AN OWN EXPERIMENT
data.org <- c(518, 157, 118, 107, # parallel, target in 1. position
              537, 148, 135, 80,  # parallel, target in 2. position
              517, 178, 127, 78,  # parallel, target in 3. position
              507, 164, 122, 107, # parallel, target in 4. position

              509, 152, 142, 97,  # serial, target in 1. position
              545, 186, 102, 67,  # serial, target in 2. position
              516, 162, 127, 95,  # serial, target in 3. position
              497, 164, 123, 121) # serial, target in 4. position

# POOL THE DATA FROM THE DIFFERENT POSITIONS
datavec <- c(data.org[1:4]   + data.org[5:8]   + data.org[9:12]  +
data.org[13:16],
             data.org[17:20] + data.org[21:24] + data.org[25:28] +
data.org[29:32])

# CONFIGRUATIONS OF THE MODELS
cfg.1 <- list(k.rg = c(4, 4), rating = F)
cfg.2 <- list(k.rg = c(4, 4), rating = F, restriction = "equal")

#============================================================
# 1. Estimation of ranking data:
#    4-AFC & 3-AFC
#    Standard UVSDT restrictions
```

[1] I would like to thank Lawrence DeCarlo for his generous support in resolving these issues and for making available his simulated data sets.

```
#============================================================
Opti.Obj <- SDT.Estimate(data = datavec, n = cfg.1, Model.Id = "SDT.Rank",
test = T)
Stat.Obj <- SDT.Statistics(Opti.Obj)      # Evaluate the results
print(Stat.Obj)

#============================================================
# 2. Estimation of ranking data:
#    4-AFC & 3-AFC
#    Equality restrictions
#============================================================
Opti.Obj <- SDT.Estimate(data = datavec, n = cfg.2, Model.Id = "SDT.Rank",
test = T)
Stat.Obj <- SDT.Statistics(Opti.Obj)      # Evaluate the results
```

The following steps are performed:

1. The data vectors `data.org` contains the frequencies for the different response categories for the different positions of the target within the distractor set. These data are pooled over positions of the target resulting in 4 data points per condition (i.e. 8 data points all in all) that are stored in the vector `datavec`. The pooling of the data is performed by the command line:

```
datavec <- c(data.org[1:4]   + data.org[5:8]   + data.org[9:12]  +
data.org[13:16], data.org[17:20] + data.org[21:24] + data.org[25:28] +
data.org[29:32])
```

2. The configuration lists for the two models are generated:

```
cfg.1 <- list(k.rg = c(4, 4), rating = F)
cfg.2 <- list(k.rg = c(4, 4), rating = F, restriction = "equal")
```

   The configuration list `cfg.1` represents the configuration of the first model:

   - `k.rg = c(4, 4)` indicates that there are two data sets with 4 data points each (=4-AFC repeated data).
   - `rating = F` indicates that only AFC data (and no rating data) are modeled.

   Due to the fact that the `restriction` option was not used, standard restrictions were set. A description of the various entries of the configuration list is presented in Chapter 3.10.

   The configuration list `cfg.2` represents the configuration of the second model:

   - `restriction = "equal"` indicates the use of equality resrtitctions.

3. The estimation procedure `SDT.Estimate()` is called using the model identification string `Model.Id = "SDT.Rank"` and the configuration list `cfg.1`. The result of the estimation is assigned to the object `Opti.Obj`.
4. Test statistics are computed by means of the function `SDT.Statistics()`, and result is assigned to the object `Stat.Obj`.
5. The content of the Object `Stat.Obj` is printed: `print(Stat.Obj)`.
6. A second run using the configuration list `cfg.2` is used for modeling the same data with the dame models and equality restrictions.

Here is the output of the two modeling runs:

```
=========================================================
 Symbolic and numeric gradients at the optimum:
 Length of difference vector  : 6.066071e-08
 Length of gradient at optimum: 9.043311e-06
=========================================================
                Symbolic   Numeric Difference
[P-1]   Mean.2  7.17e-06  7.19e-06     -1e-08
```

```
[P-1] Stddev.2 -5.34e-06 -5.31e-06     -2e-08
[P-2]   Mean.2 -1.04e-06 -1.02e-06     -2e-08
[P-2] Stddev.2  8.90e-07  8.40e-07      5e-08
=====================================================

-------------------------------------------------------------------------
Results of estimation of 4-AFC & 3-AFC data with standard restrictions:
-------------------------------------------------------------------------
$Model.description
[1] "SDT ranking model: Model types: <P,P> Number of ranking-alternatives:
<4,4> Number of ranking-positions: <4,4> Number of quadrature points: <30>
Restrictions: <standard>"

$Statistics
                                        Statistic
log L                                   -8201.526
X^2                                        12.885
G^2                                        12.719
df                                          2.000
p(Y > X^2)                                  0.002
p(Y > G^2)                                  0.002
AIC                                     16411.051
BIC                                     16438.581
CAICF                                   16472.824
ICOMP                                   16403.905
ICOMP.R                                 16403.827
Free Parameters                             4.000
Length of gradient at optimum               0.000
Rank of Hessian                             4.000
Condition number of information matrix      3.926
Rank of model matrix: t(J) * J              4.000
Condition number of model matrix            4.822

$Free.parameters
               Value    SE CFI-95(Lower) CFI-95(Upper)
[P-1]   Mean.2 1.146 0.038         1.072         1.220
[P-1] Stddev.2 1.426 0.046         1.336         1.516
[P-2]   Mean.2 1.131 0.037         1.057         1.204
[P-2] Stddev.2 1.419 0.046         1.330         1.509

$Ranking.parameters
         Rank-1[P] Rank-2[P]
Mean.1       0.000     0.000
Stddev.1     1.000     1.000
Mean.2       1.146     1.131
Stddev.2     1.426     1.419

$Data.and.Estimates
          Observed Expected Probabilities Residuals  Std.Resid.(analytical)
AFC-4 [1]     2079 2071.008         0.575     0.176                   2.901
AFC-4 [2]      647  688.579         0.191    -1.585                  -2.901
AFC-4 [3]      502  453.124         0.126     2.296                   2.901
AFC-4 [4]      372  387.289         0.108    -0.777                  -2.901
AFC-4 [1]     2067 2061.103         0.572     0.130                   2.114
AFC-4 [2]      664  694.486         0.193    -1.157                  -2.114
AFC-4 [3]      494  458.224         0.127     1.671                   2.114
AFC-4 [4]      380  391.187         0.109    -0.566                  -2.114

=====================================================
```

```
 Symbolic and numeric gradients at the optimum:
 Length of difference vector  : 2.410043e-08
 Length of gradient at optimum: 3.342198e-07
=======================================================
               Symbolic  Numeric Difference
[P-1]   Mean.2 -1.9e-07 -1.8e-07     -1e-08
[P-1] Stddev.2 -2.7e-07 -2.9e-07      2e-08
=======================================================

-------------------------------------------------------------------------
Results of estimation of 4-AFC & 3-AFC data with equality restrictions:
-------------------------------------------------------------------------
$Model.description
[1] "SDT ranking model: Model types: <P,P> Number of ranking-alternatives:
<4,4> Number of ranking-positions: <4,4> Number of quadrature points: <30>
Restrictions: <equal>"

$Statistics
                                        Statistic
log L                                   -8201.572
X^2                                        12.962
G^2                                        12.812
df                                          4.000
p(Y > X^2)                                  0.011
p(Y > G^2)                                  0.012
AIC                                     16407.144
BIC                                     16420.909
CAICF                                   16439.417
ICOMP                                   16403.571
ICOMP.R                                 16403.532
Free Parameters                             2.000
Length of gradient at optimum               0.000
Rank of Hessian                             2.000
Condition number of information matrix      3.872
Rank of model matrix: t(J) * J              2.000
Condition number of model matrix            4.736

$Free.parameters
               Value    SE CFI-95(Lower) CFI-95(Upper)
[P-1]   Mean.2 1.138 0.027         1.086         1.190
[P-1] Stddev.2 1.423 0.032         1.359         1.486

$Ranking.parameters
         Rank-1[P] Rank-2[P]
Mean.1       0.000     0.000
Stddev.1     1.000     1.000
Mean.2       1.138     1.138
Stddev.2     1.423     1.423

$Data.and.Estimates
          Observed Expected Probabilities Residuals  Std.Resid.(analytical)
AFC-4 [1]     2079 2064.617         0.574     0.317                   0.682
AFC-4 [2]      647  691.058         0.192    -1.676                  -2.253
AFC-4 [3]      502  455.358         0.126     2.186                   2.525
AFC-4 [4]      372  388.967         0.108    -0.860                  -1.239
AFC-4 [1]     2067 2067.485         0.574    -0.011                  -0.023
AFC-4 [2]      664  692.017         0.192    -1.065                  -1.432
AFC-4 [3]      494  455.990         0.126     1.780                   2.056
AFC-4 [4]      380  389.508         0.108    -0.482                  -0.694
```

1. The first table for each model fit is printed by the estimation procedure `SDT.Estimate()` since the flag `test = T` was set. It shows that the estimation procedure converged to a local optimum with the numerically and analytically computed gradients being practically identical.
2. The second part reveals for each model run the output shows the content of the object `Stat.Obj` that was compued by the function `SDT.Statistics()`.
   The output comprises the following sections:
   (i) `Model.description:` A verbal description of the model: This description exhibits the setup.
   (ii) `Statistics:` A list of various test statistics;
   (iii) `Free.parameters:` Estimated parameters with estimated standard errors and confidence intervals;
   (iv) `Ranking parameters:` A list of all model parameters including constrained parameters; The column headers indicate which model [P = parallel, S = serial] was used for the respective data set (in the present case the parallel model was used).
   (v) `Data.and.Estimates:` Observed and estimated frequencies and probabilities as well as residuals;

Comments:

- ❒ Further details concerning the output of `SDT.Statistics()` are provided in Chapter 2.2.
- ❒ The model obviously does not fit the data. In fact, the model with both data sets modeled by means of a parallel model fits the data much better.
- ❒ The code for modeling these data is contained in the file:
  `SDT-Rank (Ex.2 Modeling of pure ranking data).R`

13.1.2 SDT-Rank Example 2: Joint modeling repeated forced choice and rating data

The present example demonstrates the utilization of the `SDT.Rank` module for fitting repeated forced choice together with rating data. The data come from an own experiment. They comprise data from a 4-AFC and 3-AFC task with repeated choices and two sets of rating data with 6 and 4 response categories. The four data sets are modelled together assuming equal paraemters for all 4 data sets.

Here is an extract of R code for modeling the two sets 4-AFC data:

```
data.pooled <- c(10198, 3209, 2252, 1621,              # 4-AFC
                 11382, 3519, 2379,                    # 3-SFC
                 4611,  4340, 3357, 2663, 1618,  691,  # 6-Rating (New)
                 1309,  2049, 2274, 2620, 2493, 6535,  # 6 Rating (Old)
                 5508,  6711, 3987, 1074,              # 4-Rating (New)
                 1837,  3959, 4330, 7154)              # 4-Rating (old)

# CONFIGURATION LIST
cfg <- list(n.sdt = c(2, 6, 2, 4), k.rg = c(4, 3), rating = T, restriction
= "equal")

# ESTIMATE AND EVALUATE RESULTS
Opti.Obj <- SDT.Estimate(data = data.pooled, n = cfg, Model.Id =
"SDT.Rank", test = T)
Stat.Obj <- SDT.Statistics(Opti.all.equ.Obj, display.warning = F)

# PRINT THE RESULTS
print(Stat.Obj)1
```

This piece of code comprises the following complonents:

1. The vector `data.pooled` contains the data in the proper order:
   - ❒ The data from the repeated 4-AFC and 3.AFC tasks.
   - ❒ The data from the rating task with 6 response categories (and two types of signals)
   - ❒ The data from the rating task with 4 response categories (and two types of signals)
2. The configuration list:

```
cfg <- list(n.sdt = c(2, 6, 2, 4), k.rg = c(4, 3), rating = T, restriction
= "equal")
```

   provides the following information:
   The specification `n.sdt = c(2, 6, 2, 4)` tells the program that two rating models are used:
   - ❒ The first data set has two types of stimuli (and the model has thus two Gaussian distributions) and 6 response categpries.
   - ❒ The second data set comprises two types of stimuli and 4 response categpries.

   The specification `k.rg = c(4, 3)` informs the model that 4-AFC and 3-AFC data with repeated forced choices are present.

   Finally, the specification `restriction = "equal"` tells the program to set equality restrictions.
3. The following two commands result in fitting the model and evaluating the results:

```
Opti.Obj <- SDT.Estimate(data = data.pooled, n = cfg, Model.Id =
"SDT.Rank", test = T)
Stat.Obj <- SDT.Statistics(Opti.all.equ.Obj, display.warning = F)
```

4. Finally, the result is printed.

This results in the following output:

•••

```
$Model.description
[1] "SDT ranking and rating model: Number of Gaussians within SDT models: <<2,6>,
<2,4>> Model types: <P,P> Number of ranking-alternatives: <4,3> Number of ranking-
positions: <4,3> Number of quadrature points: <30> Restrictions: <equal>"

$Statistics
                                        Statistic
log L                                -135214.832
X^2                                     208.109
G^2                                     209.173
df                                       11.000
p(Y > X^2)                                0.000
p(Y > G^2)                                0.000
AIC                                  270449.664
BIC                                  270545.154
CAICF                                270662.847
ICOMP                                270434.156
ICOMP.R                              270433.883
Free Parameters                          10.000
Length of gradient at optimum             0.001
Rank of Hessian                          10.000
Condition number of information matrix    20.933
Rank of model matrix: t(J) * J           10.000
Condition number of model matrix         25.973

$Free.parameters
                 Value    SE CFI-95(Lower) CFI-95(Upper)
[P-1]   Mean.2   1.208 0.009         1.191         1.225
[P-1] Stddev.2   1.369 0.009         1.351         1.388
[SDT-1]      c-1 -0.652 0.009        -0.670        -0.634
```

```
[SDT-1]      c-2  0.044 0.008         0.028         0.060
[SDT-1]      c-3  0.575 0.008         0.559         0.592
[SDT-1]      c-4  1.120 0.009         1.101         1.138
[SDT-1]      c-5  1.666 0.012         1.643         1.689
[SDT-2]      c-1 -0.477 0.009        -0.495        -0.460
[SDT-2]      c-2  0.577 0.008         0.560         0.593
[SDT-2]      c-3  1.508 0.011         1.487         1.530

$Ranking.parameters
         Rank-1[P] Rank-2[P]
Mean.1       0.000     0.000
Stddev.1     1.000     1.000
Mean.2       1.208     1.208
Stddev.2     1.369     1.369

$Rating.parameters
       SDT-1/Gauss-1 SDT-1/Gauss-2 SDT-2/Gauss-1 SDT-2/Gauss-2
Mean               0         1.208             0         1.208
Stddev             1         1.369             1         1.369

$Decision.bounds
         c-1   c-2   c-3  c-4   c-5
SDT-1 -0.652 0.044 0.575 1.12 1.666
SDT-2 -0.477 0.577 1.508

$Data.and.Estimates
            Observed  Expected Probabilities Residuals  Std.Resid.(analytical)
AFC-4 [1]      10198 10248.347         0.593    -0.497                  -0.906
AFC-4 [2]       3209  3332.108         0.193    -2.133                  -2.563
AFC-4 [3]       2252  2084.142         0.121     3.677                   4.071
AFC-4 [4]       1621  1615.403         0.093     0.139                   0.165
AFC-3 [1]      11382 11359.050         0.657     0.215                   0.416
AFC-3 [2]       3519  3610.833         0.209    -1.528                  -1.844
AFC-3 [3]       2379  2310.117         0.134     1.433                   1.714
SDT-1.1 [1]     4611  4443.832         0.257     2.508                   6.956
SDT-1.1 [2]     4340  4500.661         0.260    -2.395                  -5.249
SDT-1.1 [3]     3357  3454.294         0.200    -1.655                  -3.187
SDT-1.1 [4]     2663  2610.929         0.151     1.019                   1.706
SDT-1.1 [5]     1618  1442.971         0.084     4.608                   6.615
SDT-1.1 [6]      691   827.313         0.048    -4.739                  -7.016
SDT-1.2 [1]     1309  1507.182         0.087    -5.105                  -7.116
SDT-1.2 [2]     2049  1910.256         0.111     3.174                   4.158
SDT-1.2 [3]     2274  2149.644         0.124     2.682                   3.815
SDT-1.2 [4]     2620  2631.200         0.152    -0.218                  -0.351
SDT-1.2 [5]     2493  2705.517         0.157    -4.086                  -7.524
SDT-1.2 [6]     6535  6376.200         0.369     1.989                   4.448
SDT-2.1 [1]     5508  5470.385         0.317     0.509                   1.427
SDT-2.1 [2]     6711  6934.476         0.401    -2.684                  -6.513
SDT-2.1 [3]     3987  3739.037         0.216     4.055                   7.307
SDT-2.1 [4]     1074  1136.102         0.066    -1.842                  -2.879
SDT-2.2 [1]     1837  1888.180         0.109    -1.178                  -1.659
SDT-2.2 [2]     3959  3683.610         0.213     4.537                   6.873
SDT-2.2 [3]     4330  4569.110         0.264    -3.537                  -6.597
SDT-2.2 [4]     7154  7139.099         0.413     0.176                   0.390
```

The output contains the following sections:

1. `$Model.description`: A string showing the confirguration of the model.
2. `$Free.parameters`: The free parametes with estimated standard errors and confidence intervals.
3. `$Ranking.parameters`: The parameters of the two AFC (or ranking) models.
4. `$Rating.parameters`: The Gaussian parameters of the two rating models.
5. `$Decision.bounds`: The decision thresholds for the two rating models.

6. `$Data.and.Estimates`: The observed and estimated frequencies and the estimated probabilities as well as the residuals.

Comment: The code fo this example is contained in the file:
`SDT-Rank (Ex.5 Own data).R`

13.1.3 Files with SDT-Rank Examples

The following files contain examples of the `SDT.Rank` model:
```
SDT-Rank (Ex.1 Parameterinformation).R
SDT-Rank (Ex.2 Modeling of pure ranking data).R
SDT-Rank (Ex.3 Kellen, Klauer & Singmann, 2012).R
SDT-Rank (Ex.4 Model only first choice with k alternatives).R
SDT-Rank (Ex.5 Own data).R
```

14. Working Examples: SDT model with recollection and guessing for modeling forced choice and rating data with or without bias (HTSDT.Rank & HTSDT.Bias.Rank)

In the following examples of applications of the module `HTSDT.Rank` and `HTSDT.Bias.Rank` are presented. First, two examples illustrating the application of `HTSDT.Rank` are presented. This is followed by one example illustrating the application of `HTSDT.Bias.Rank`.

14.1 HTSDT.Rank examples

The first example demonstrates the simultaneous modeling of forced choice and rating data.

14.1.1 HTSDT-Rank Example 1: Modeling forced choice and rating data with standard restrictions

The code of this example is contained in the file:
`HTSDT-Rank (Ex.2 Kellen, Klauer & Singmann, 2012).R`

The following R script demonstrates the estimation of both rating and forced choice data from Kellen et al. (2012) that takes recollection into account. This can be performed by using the standard restrictions.

```
# Data vector of Kellen, Klauer & Singmann (2012), pooled data
datavec <- c(1801, 488, 407, 304,                 # 1. Ranking data
             667, 780, 651, 485, 277,  140,       # 2. Rating data (new)
             198, 303, 402, 421, 426, 1250)       # 3. Rating data (old)

cfg <- list(n.sdt = c(2, 6), k.rg = 4, rating = T, restriction = "standard")
Opti.Obj <- SDT.Estimate(data = datavec, n = cfg, Model.Id = "HTSDT.Rank", test =
T)
Stat.Obj <- SDT.Statistics(Opti.Obj)
------------------------------------------------------------------------
Model 1:  HTSDT-Modell AFC + rating
          standard restrictions
          AFC: k.rg = 4
          SDT: c(2, 6)
------------------------------------------------------------------------
print(Stat.Obj)
```

The only difference to the code presented in Chapter 13 consists in the usage of a different model identification string passed to the function `SDT.Estimate()`: Instead of `"SDT.Rank"` the string `"HTSDT.Rank"` is used indicating the usage of the HTSDT-Rank model.
Here is a selection of the output produced by the R script:

```
------------------------------------------------------------------------
Model 1:  HTSDT-Modell AFC + rating
          standard restrictions
          AFC: k.rg = 4
          SDT: c(2, 6)
```

```
------------------------------------------------------------------------------
$Model.description
[1] "HTSDT ranking and rating model: Number of Gaussians within SDT models: <2,6>
Model types: <P> Number of ranking-alternatives: <4> Number of ranking-positions:
<4> Number of quadrature points: <30> Restrictions: <standard>"

$Statistics
                                      Statistic
log L                                -13135.511
X^2                                     13.889
G^2                                     13.802
df                                       4.000
p(Y > X^2)                               0.008
p(Y > G^2)                               0.008
AIC                                  26289.021
BIC                                  26352.966
CAICF                                26434.694
ICOMP                                26280.375
ICOMP.R                              26275.826
Free Parameters                          9.000
Length of gradient at optimum            0.000
Rank of Hessian                          9.000
Condition number of information matrix  111.375
Rank of model matrix: t(J) * J           9.000
Condition number of model matrix       150.824

$Free.parameters
                              Value    SE CFI-95(Lower) CFI-95(Upper)
Recollection (Rank 1)         0.385 0.021         0.343         0.426
Mean.2       (Rank 1)         0.360 0.056         0.250         0.470
[SDT-1][Gauss-2] Recollection 0.315 0.071         0.176         0.454
[SDT-1][Gauss-2]         Mean 0.644 0.040         0.566         0.723
[SDT-1]      c-1             -0.736 0.024        -0.783        -0.689
[SDT-1]      c-2             -0.049 0.022        -0.092        -0.007
[SDT-1]      c-3              0.507 0.022         0.464         0.551
[SDT-1]      c-4              1.049 0.025         1.000         1.098
[SDT-1]      c-5              1.687 0.039         1.610         1.765

$Ranking.parameters
             Rank <k=4, j=4>
Recollection           0.385
Mean.1                 0.000
Stddev.1               1.000
Mean.2                 0.360
Stddev.2               1.000

$Rating.parameters
             SDT-1/Gauss-1 SDT-1/Gauss-2
Recollection             0         0.315
Mean                     0         0.644
Stddev                   1         1.000

$Decision.bounds
         c-1    c-2   c-3   c-4   c-5
SDT-1 -0.736 -0.049 0.507 1.049 1.687

$Data.and.Estimates
            Observed Expected Probabilities Residuals  Std.Resid.(analytical)
AFC-4 [1]       1801 1801.000         0.600     0.000                   0.000
AFC-4 [2]        488  492.861         0.164    -0.219                  -0.558
AFC-4 [3]        407  397.940         0.133     0.454                   0.558
AFC-4 [4]        304  308.199         0.103    -0.239                  -0.558
SDT-1.1 [1]      667  692.660         0.231    -0.975                  -3.452
SDT-1.1 [2]      780  748.161         0.249     1.164                   2.523
SDT-1.1 [3]      651  641.028         0.214     0.394                   0.734
SDT-1.1 [4]      485  476.636         0.159     0.383                   0.694
```

```
SDT-1.1 [5]      277  304.160         0.101    -1.557                  -2.770
SDT-1.1 [6]      140  137.356         0.046     0.226                   3.219
SDT-1.2 [1]      198  172.119         0.057     1.973                   3.402
SDT-1.2 [2]      303  329.136         0.110    -1.441                  -2.158
SDT-1.2 [3]      402  414.147         0.138    -0.597                  -0.887
SDT-1.2 [4]      421  434.878         0.145    -0.665                  -1.127
SDT-1.2 [5]      426  399.720         0.133     1.314                   2.862
SDT-1.2 [6]     1250 1250.000         0.417     0.000                   0.000
```

The output includes a new section labeled `$Ranking.parameters` that lists the parameters of the forced choice component. In case of multiple forced choice models a column is provided for each forced choice model (cf. Chapter 14.1.2). Inspection of the results exhibits, e.g., that the estimated value of the recollection parameter is .385.
The labels in the section `$Data.and.Estimates` indicate whicht data are forced choice data (`AFC-4`) and which are new (`SDT 1.1`) and old items (`SDT 1.2`), respectively.

Comment:
The file `HTSDT-Rank (Ex.2 Kellen, Klauer & Singmann, 2012).R` contains an example that uses equality restrictions (`restriction = "equal"`).

14.1.2 HTSDT-Rank Example 2: Modeling multiple forced choice data

The second example demonstrates the modeling of multiple forced choice data using UVSDT restrictions (for a detailed description of this option, cf. Chapter 3.11.1). The code of this example is contained in the file:

`HTSDT-Rank (Ex.3 Multiple forced choice models).R`

The present example also illustrates the simultaneous modeling of two sets of forced choice data, one from a 2-AFC task and the other one from a 4-AFC task. The relevant piece of R code is as follows:

```
dvec <- c(2327, 873,                          # 1. 2-AFC data
          739, 351, 295, 215)                 # 2. 4-AFC data

cfg <- list(j.rg = c(2, 4), rating = F, restriction = "UVSDT")
fixed <- matrix(c(1, 5), nr = 2)   # Standard deviation of first model to 1
Opti.Obj <- SDT.Estimate(data = dvec, n = cfg, Model.Id = "HTSDT.Rank", test = T)
Stat.Obj <- SDT.Statistics(Opti.Obj)           # Evaluate the results
print(Stat.Obj)
```

The data vector `dvec` comprises 6 data points, 2 from the 2-AFC task and 4 from the 4-AFC task. In addition. in the configuration list `cfg` the vector `c(2, 4)` is passed to argument `j.rg` indicating two models with 2 and 4 data points respectively. The flag `rating` was set to `FALSE` indicating that forced choice data are modeled only. The string passed to the argument `restriction` was `"UVSDT"` indicating restictions that result in the UVSDT model. Specifically, the recollection probability is set to zero.
For identification purposes, the and standard deviation parameter of the old distribution for the model used to model the 2-AFC data was fixed to 1.
Here is the output:

```
------------------------------------------------------
 2. Modeling (repeated) AFC data (2-AFC and 4-AFC)
     Additional restriction:
    - Standard deviation of first model = 1
------------------------------------------------------
$Model.description
[1] "HTSDT ranking model: Model types: <P,P> Number of ranking-alternatives: <2,4>
Number of ranking-positions: <2,4> Number of quadrature points: <30> Restrictions:
<UVSDT>"

$Statistics
                                        Statistic
```

```
log L                                      -3910.825
X^2                                            3.816
G^2                                            3.781
df                                             1.000
p(Y > X^2)                                     0.051
p(Y > G^2)                                     0.052
AIC                                         7827.649
BIC                                         7847.078
CAICF                                       7872.143
ICOMP                                       7822.056
ICOMP.R                                     7821.808
Free Parameters                                3.000
Length of gradient at optimum                  0.000
Rank of Hessian                                3.000
Condition number of information matrix         3.137
Rank of model matrix: t(J) * J                 3.000
Condition number of model matrix               2.899

$Free.parameters
                      Value    SE CFI-95(Lower) CFI-95(Upper)
Mean.2       (Rank 1) 0.855 0.034         0.789         0.920
Mean.2       (Rank 2) 0.692 0.042         0.609         0.775
Stddev.2     (Rank 2) 1.208 0.055         1.099         1.316

$Ranking.parameters
             Rank <k=2, j=2> Rank <k=4, j=4>
Recollection           0.000           0.000
Mean.1                 0.000           0.000
Stddev.1               1.000           1.000
Mean.2                 0.855           0.692
Stddev.2               1.000           1.208

$Data.and.Estimates
          Observed Expected Probabilities Residuals  Std.Resid.(analytical)
AFC-2 [1]     2327 2327.000         0.727     0.000                   0.000
AFC-2 [2]      873  873.000         0.273     0.000                   0.000
AFC-4 [1]      739  733.904         0.459     0.188                   1.953
AFC-4 [2]      351  373.025         0.233    -1.140                  -1.953
AFC-4 [3]      295  270.486         0.169     1.491                   1.953
AFC-4 [4]      215  222.586         0.139    -0.508                  -1.953
```

There are two things to notice: First, the section `$Ranking.parameters`, showing the parameters of the forced choice models, now contains two columns for the two models. Due to the fact that no rating data were modeled there is no section containing s parameters of the model for modeling rating data.
Second, The row labels in the section `$Data.and.Estimates` indicate which data points and estimates etc. belong to the first (`AFC-2`) and to the second data set (`AFC-4`), respectively.

Comment:
The file `HTSDT-Rank (Ex.3 Multiple forced choice models).R` contains additional code for estimating the data with standard and equality restrictions.

14.1.3 HTSDT-Rank Example 3: Modeling multiple rating and multiple forced choice data simultaneously

The third example demonstrates the simultaneous modeling of multiple forced choice and multiple rating data with equality restrictions (for a detailed description of this option, cf. Chapter 3.11.1). The code of this example is contained in the file:

`HTSDT-Rank (Ex.4 Own data).R`

The relevant piece of R code is as follows:

```
data.pooled <- c(10198, 3209, 2252, 1621,              # 4-AFC
                 11382, 3519, 2379,                    # 3-SFC
```

```
                4611,  4340, 3357, 2663, 1618,  691,  # 6-Rating (New)
                1309,  2049, 2274, 2620, 2493, 6535,  # 6 Rating (Old)
                5508,  6711, 3987, 1074,              # 4-Rating (New)
                1837,  3959, 4330, 7154)              # 4-Rating (Old)

cfg.all.equal <- list(n.sdt = c(2, 6, 2, 4), k.rg = c(4, 3), rating = T,
restriction = "equal")
Opti.all.equ.Obj <- SDT.Estimate(data = data.pooled, n = cfg.all.equal, Model.Id =
"HTSDT.Rank", test = T)
Stat.all.equ.Obj <- SDT.Statistics(Opti.all.equ.Obj, display.warning = F)
print(Stat.all.equ.Obj)
```

The entries in the configuration list,

```
list(n.sdt = c(2, 6, 2, 4), k.rg = c(4, 3), rating = T, restriction = "equal")
```

have the following meaning:

```
n.sdt = c(2, 6, 2, 4)
```

specifies the rating model and the number of response categories. In the present case, there are two models: The first model comprises two types of signals with 6 response categories for each signal. The second model comprises also two types of signals, however, with only four response categories.

```
k.rg = c(4, 3)
```

specifies the repeated forced choice models: a 4-AFC and a 3-AFC model with repeated choices.

```
rating = T
```

informs the estimation function that rating as well as AFC (or ranking) data are used. Finally,

```
restriction = "equal"
```

specifies the type of restrictions. In the present case the recollection and mean parameters are all assumed to be equal.

This piece of code results in the following output:

```
$Model.description
[1] "HTSDT ranking and rating model: Number of Gaussians within SDT models: <<2,6>,
<2,4>> Model types: <P,P> Number of ranking-alternatives: <4,3> Number of ranking-
positions: <4,3> Number of quadrature points: <30> Restrictions: <equal>"

$Statistics
                                          Statistic
log L                                    -135139.835
X^2                                          59.278
G^2                                          59.180
df                                           11.000
p(Y > X^2)                                    0.000
p(Y > G^2)                                    0.000
AIC                                      270299.671
BIC                                      270395.162
CAICF                                    270510.584
ICOMP                                    270285.836
ICOMP.R                                  270283.790
Free Parameters                              10.000
Length of gradient at optimum                 0.000
Rank of Hessian                              10.000
Condition number of information matrix        35.759
Rank of model matrix: t(J) * J               10.000
Condition number of model matrix             69.439

$Free.parameters
                      Value    SE CFI-95(Lower) CFI-95(Upper)
Recollection (Rank 1)  0.292 0.004         0.284         0.300
Mean.2       (Rank 1)  0.600 0.010         0.581         0.618
[SDT-1]      c-1      -0.631 0.009        -0.649        -0.614
```

```
[SDT-1]      c-2       0.027 0.008         0.011         0.042
[SDT-1]      c-3       0.533 0.008         0.517         0.549
[SDT-1]      c-4       1.082 0.009         1.063         1.100
[SDT-1]      c-5       1.763 0.016         1.733         1.794
[SDT-2]      c-1      -0.460 0.009        -0.477        -0.444
[SDT-2]      c-2       0.539 0.008         0.523         0.555
[SDT-2]      c-3       1.539 0.013         1.514         1.565

$Ranking.parameters
            Rank <k=4, j=4> Rank <k=3, j=3>
Recollection           0.292           0.292
Mean.1                 0.000           0.000
Stddev.1               1.000           1.000
Mean.2                 0.600           0.600
Stddev.2               1.000           1.000

$Rating.parameters
            SDT-1/Gauss-1 SDT-1/Gauss-2 SDT-2/Gauss-1 SDT-2/Gauss-2
Recollection             0         0.292             0         0.292
Mean                     0         0.600             0         0.600
Stddev                   1         1.000             1         1.000

$Decision.bounds
         c-1   c-2   c-3   c-4   c-5
SDT-1 -0.631 0.027 0.533 1.082 1.763
SDT-2  -0.46 0.539 1.539

$Data.and.Estimates
            Observed  Expected Probabilities Residuals  Std.Resid.(analytical)
AFC-4 [1]      10198 10234.738         0.592    -0.363                  -0.650
AFC-4 [2]       3209  3261.338         0.189    -0.916                  -1.078
AFC-4 [3]       2252  2281.808         0.132    -0.624                  -0.693
AFC-4 [4]       1621  1502.115         0.087     3.067                   3.515
AFC-3 [1]      11382 11321.851         0.655     0.565                   1.080
AFC-3 [2]       3519  3695.431         0.214    -2.902                  -3.423
AFC-3 [3]       2379  2262.718         0.131     2.445                   2.882
SDT-1.1 [1]     4611  4560.728         0.264     0.744                   1.748
SDT-1.1 [2]     4340  4262.010         0.247     1.195                   2.383
SDT-1.1 [3]     3357  3326.234         0.192     0.533                   0.946
SDT-1.1 [4]     2663  2716.400         0.157    -1.025                  -1.709
SDT-1.1 [5]     1618  1741.761         0.101    -2.965                  -4.594
SDT-1.1 [6]      691   672.867         0.039     0.699                   1.646
SDT-1.2 [1]     1309  1336.603         0.077    -0.755                  -1.031
SDT-1.2 [2]     2049  2131.270         0.123    -1.782                  -2.430
SDT-1.2 [3]     2274  2328.851         0.135    -1.137                  -1.622
SDT-1.2 [4]     2620  2589.122         0.150     0.607                   0.957
SDT-1.2 [5]     2493  2357.819         0.136     2.784                   4.948
SDT-1.2 [6]     6535  6536.335         0.378    -0.017                  -0.032
SDT-2.1 [1]     5508  5574.245         0.323    -0.887                  -2.122
SDT-2.1 [2]     6711  6607.153         0.382     1.278                   2.713
SDT-2.1 [3]     3987  4029.792         0.233    -0.674                  -1.258
SDT-2.1 [4]     1074  1068.810         0.062     0.159                   0.331
SDT-2.2 [1]     1837  1769.547         0.102     1.603                   2.278
SDT-2.2 [2]     3959  4053.641         0.235    -1.486                  -2.244
SDT-2.2 [3]     4330  4291.930         0.248     0.581                   1.056
SDT-2.2 [4]     7154  7164.882         0.415    -0.129                  -0.246
```

Note that the mean and recollection paramters of all models (AFC and rating) are equal due to the equality constraints. The standard deviations parameters are all fixed to one. Note also that there are two sets of thresholds for the two rating models.

14.1.4 Files with HTSDT-Rank Examples

The following files contain examples of the `HTSDT.Rank` model:

```
HTSDT-Rank (HTSDT-Rank (Ex.1 Parameters & restrictions I AFC & Rating).R
HTSDT-Rank (Ex.2 Kellen, Klauer & Singmann, 2012).R
```

```
HTSDT-Rank (Ex.3 Multiple forced choice models).R
HTSDT-Rank (Ex.4 Own data).R
```

14.2 HTSDT-Bias-Rank example

The following example demonstrates the application of the `HTSDT.Bias.Rank` model to a data set comprising two sets of 4-AFC data as well as rating data (from an own experiment). The code of this example is contained in the file:

```
MIX-Bias-Rank (Ex.4 Estimate Ranking and rating model (multiple models)).R
```

The relevant code for fitting the data, using standard restrictions, and evaluating as well as printing the results looks like this:

```
datavec <- c(210,  68,  68,  54,     # Ranking 1, Target on Position 1
             193,  98,  60,  49,     # Ranking 1, Target on Position 2
             198, 101,  55,  46,     # Ranking 1, Target on Position 3
             169,  88,  82,  61,     # Ranking 1, Target on Position 4
             180,  79,  78,  63,     # Ranking 2, Target on Position 1
             184,  86,  85,  45,     # Ranking 2, Target on Position 2
             190, 100,  58,  52,     # Ranking 2, Target on Position 3
             185,  86,  74,  55,     # Ranking 2, Target on Position 4
             432, 341, 309, 262, 185,  71,  # Rating data (new)
             179, 220, 238, 256, 257, 450)  # Rating data (old)
cfg <- list(k.rg = c(4, 4), j.rg = c(4, 4))
Opti1.Obj <- SDT.Estimate(data = datavec, n = cfg, Model.Id =
"HTSDT.Bias.Rank", test = T)
Stat1.Obj <- SDT.Statistics(Opti1.Obj)    # Evaluate the results
print(Stat1.Obj)
```

Here is a portion of the output:

```
$Model.description
[1] "SDT ranking and rating model with recollection and guessing, with
position bias: Number of Gaussian models: <2> Number of ranking-
alternatives: <4, 4> Number of ranking-positions: <4, 4> Number of
quadrature points: <30> Type of restrictions: <standard>"

$Statistics
                                      Statistic
log L                                 -9519.546
X^2                                      28.839
G^2                                      28.748
df                                       21.000
p(Y > X^2)                                0.118
p(Y > G^2)                                0.120
AIC                                   19065.091
BIC                                   19153.024
CAICF                                 19259.752
ICOMP                                 19049.841
ICOMP.R                               19044.094
Free Parameters                          13.000
Length of gradient at optimum             0.000
Rank of Hessian                          13.000
Condition number of information matrix   108.959
Rank of model matrix: t(J) * J           13.000
Condition number of model matrix         51.337

$Free.parameters
                          Value    SE CFI-95(Lower) CFI-95(Upper)
Recollection    (Rank 1)  0.184 0.014         0.157         0.212
Mean.2          (Rank 1)  0.410 0.032         0.346         0.473
```

```
Bias.1          (Rank 1)  0.173 0.087         0.002         0.344
Bias.2          (Rank 1)  0.185 0.085         0.020         0.351
Bias.3          (Rank 1)  0.230 0.084         0.064         0.395
Bias.1          (Rank 2) -0.077 0.088        -0.249         0.095
Bias.2          (Rank 2)  0.028 0.086        -0.140         0.195
Bias.3          (Rank 2)  0.078 0.086        -0.090         0.246
c[1]                     -0.638 0.029        -0.694        -0.582
c[2]                     -0.063 0.026        -0.114        -0.012
c[3]                      0.427 0.026         0.375         0.478
c[4]                      0.951 0.030         0.893         1.009
c[5]                      1.668 0.051         1.567         1.768

$Ranking.parameters
             Rank <k=4, j=4> Rank <k=4, j=4>
Recollection           0.184           0.184
Guessing               0.000           0.000
Mean.1                 0.000           0.000
Stddev.1               1.000           1.000
Mean.2                 0.410           0.410
Stddev.2               1.000           1.000
Bias.1                 0.173          -0.077
Bias.2                 0.185           0.028
Bias.3                 0.230           0.078
Bias.4                 0.000           0.000

$Rating.parameters
                 Gauss-1 Gauss-2
Recollection       0.000   0.184
Guessing           0.000   0.000
Bias (guessing)    0.500   0.500
Mean               0.000   0.410
Stddev             1.000   1.000
t-1               -0.638  -0.638
t-2               -0.063  -0.063
t-3                0.427   0.427
t-4                0.951   0.951
t-5                1.668   1.668
[...]
```

The output is quite similar to that of the `HTSDT.Rank` model (cf. Chapter 14.1). The main difference consists in the fact that in the section `$Ranking.parameters` listing the parameters of the forced choice models position bias parameters are shown.
The values of the bias parameters indicate that for the first forced choice data set position 4 received less attention than the other 3 positions (these can be seen directly from the data set where the target item was ranked considerably less often as the first item, when it was on the fourth position).
For the second forced choice data set the third position received slightly more weight than the other positions. This is also evident from the frequencies since the target item was ranked more often on the first and second rank when it was on the third position.

Comments:

- The file `HTSDT-Bias-Rank (Ex.4 Estimate Ranking and rating model (multiple models)).R` contains additional code for estimating the data with extended (corresponding to the unequal variance SDT model with recollection) as well as SDT restrictions (corresponding to the unequal variance model without recollection).
- The following files contain examples of the `HTSDT.Bias.Rank` model:

```
HTSDT-Bias-Rank (Ex.0 Parameters & restrictions).R
HTSDT-Bias-Rank (Ex.1 Estimate Ranking model (1 model only)).R
HTSDT-Bias-Rank (Ex.3 Estimate Ranking model (multiple models).R
HTSDT-Bias-Rank (Ex.4 Estimate Ranking and rating model (multiple
models)).R
```

15. Working Example: Gaussian mixture SDT model for modeling forced choice and rating data with and without position bias (MIX.Rank & MIX.Bias.Rank)

In the following, examples of applications of the module `MIX.Rank` and `MIX.Bias.Rank` are presented. First, two examples illustrating the application of `MIX.Rank` are presented. This is followed by one example illustrating the application of `MIX.Bias.Rank`.

15.1 MIX.Rank Examples

The first example demonstrates the simultaneous modeling of forced choice and rating data using again the data of Kellen et al. (2012).

15.1.1 MIX-Rank Example 1: Modeling forced choice and rating data

The code of this example is contained in the file:

```
MIX-Rank (Ex.1 Kellen, Klauer & Singmann, 2012).R
```

The following R script demonstrates the estimation of both rating and forced choice data from Kellen et al. (2012) using the mixture model with extended restrictions, i.e., the mean of the weak target distribution is a free parameter (and not 0.0 as is the case with standard restrictions).

```
# Data vector of Kellen, Klauer & Singmann (2012), pooled data
datavec <- c(1801, 488, 407, 304,                 # 1. Ranking data
             667, 780, 651, 485, 277,  140,       # 2. Rating data (new)
             198, 303, 402, 421, 426, 1250)       # 3. Rating data (old)

cfg <- list(k.rg = 4, rating = T, restriction = "extended")

Opti.Obj <- SDT.Estimate(data = datavec, n = cfg, Model.Id = "MIX.Rank", test = T)
Stat.Obj <- SDT.Statistics(Opti.Obj)
print(Stat.Obj)
```

The only difference to the code presented in Chapters 13 and 14 consists in the usage of a different model identification string passed to the function `SDT.Estimate()`: Instead of the string `"SDT.Rank"` (and `"HTSDT.Rank"`, respectively) `"MIX.Rank"` is used indicating the usage of the MIX-Rank model.

Here is a selection of the output produced by the R script:

```
$Model.description
[1] "MIX ranking and rating model: Number of Mixture models: <2> Number of ranking-
alternatives: <4> Number of ranking-positions: <4> Number of quadrature points:
<30> Type of restrictions: <extended>"

$Statistics
                                      Statistic
log L                                -13137.607
X^2                                     17.925
G^2                                     17.995
df                                       5.000
p(Y > X^2)                               0.003
p(Y > G^2)                               0.003
AIC                                  26291.214
BIC                                  26348.054
CAICF                                26420.167
ICOMP                                26294.392
ICOMP.R                              26282.671
Free Parameters                          8.000
Length of gradient at optimum            0.000
```

```
Rank of Hessian                              8.000
Condition number of information matrix      736.698
Rank of model matrix: t(J) * J               8.000
Condition number of model matrix          1171.838

$Free.parameters
                    Value    SE CFI-95(Lower) CFI-95(Upper)
p.mix    (Rank 1)   0.469 0.044         0.384         0.555
Mean.1   (Rank 1)   0.338 0.075         0.191         0.485
Mean.2   (Rank 1)   2.451 0.198         2.063         2.840
c[1]               -0.779 0.024        -0.826        -0.733
c[2]               -0.078 0.021        -0.119        -0.037
c[3]                0.504 0.022         0.460         0.547
c[4]                1.078 0.026         1.026         1.129
c[5]                1.684 0.037         1.612         1.756

$Ranking.parameters
         Rank <k=4, j=4>
p.mix              0.469
Mean.0             0.000
Stddev.0           1.000
Mean.1             0.338
Stddev.1           1.000
Mean.2             2.451
Stddev.2           1.000

$Rating.parameters
         Gauss-1 Gauss-2
p.mix      1.000   0.469
Mean.0     0.000   0.338
Stddev.0   1.000   1.000
Mean.1     0.000   2.451
Stddev.1   1.000   1.000
t-1       -0.779  -0.779
t-2       -0.078  -0.078
t-3        0.504   0.504
t-4        1.078   1.078
t-5        1.684   1.684

[...]
```

As for the HTSDT-Rank model (cf Chapter 14.1), the output includes a new section labeled `$Ranking.parameters` that lists the parameters of the forced choice component. In case of multiple forced choice models a column for each model is provided (cf. Chapter 15.1.2).
Inspection of the results exhibits, e.g., that the estimated value of the mixture probability parameter is .469 (which was forced to be equal to the respective parameter of the SDT component representing the target items).

Comment:
The file `MIX-Rank (Ex.1 Kellen, Klauer & Singmann, 2012).R` contains additional code for estimating the data with standard mixture model as well as SDT restrictions (corresponding to the unequal variance model without recollection).

15.1.2 MIX-Rank Example 2: Modeling multiple forced choice data

The second example demonstrates the modeling of multiple forced choice data using standard restrictions (for a detailed description of the different types of restrictions cf. Chapter 3.12.1). The code of this example is contained in the file:

`MIX-Rank (Ex.2 Multiple forced choice models).R`

The relevant piece of R code is as follows:

```
dvec <- c(2327, 873,                          # 1. 2-AFC data
```

```
           739, 351, 295, 215)              # 2. 4-AFC data

cfg <- list(j.rg = c(2, 4), rating = F)
Opti.Obj <- SDT.Estimate(data = dvec, n = cfg, Model.Id = "MIX.Rank", test = T)
Stat.Obj <- SDT.Statistics(Opti.Obj)          # Evaluate the results
print(Stat.Obj)
```

The data vector `dvec` comprises 6 data points, 2 from the 2-AFC task and 4 from the 4-AFC task. In addition. in the configuration list `cfg` the vector `c(2, 4)` is passed to argument `j.rg` indicating two models with 2 and 4 data points respectively. The flag `rating` was set to `FALSE` indicating that forced choice data are modeled only.

Here is the output:

```
$Model.description
[1] "MIX ranking model: Number of ranking-alternatives: <2, 4> Number of ranking-
positions: <2, 4> Number of quadrature points: <30> Type of restrictions:
<standard>"

$Statistics
                                        Statistic
log L                                   -3922.173
X^2                                        26.802
G^2                                        26.478
df                                          2.000
p(Y > X^2)                                  0.000
p(Y > G^2)                                  0.000
AIC                                      7848.346
BIC                                      7861.299
CAICF                                    7874.068
ICOMP                                    7846.909
ICOMP.R                                  7846.683
Free Parameters                             2.000
Length of gradient at optimum               0.000
Rank of Hessian                             2.000
Condition number of information matrix      49.892
Rank of model matrix: t(J) * J              2.000
Condition number of model matrix           66.671

$Free.parameters
                 Value    SE CFI-95(Lower) CFI-95(Upper)
p.mix    (Rank 1) 0.626 0.060         0.509         0.743
Mean.2   (Rank 1) 1.299 0.157         0.991         1.607

$Ranking.parameters
         Rank <k=2, j=2> Rank <k=4, j=4>
p.mix              0.626           0.626
Mean.0             0.000           0.000
Stddev.0           1.000           1.000
Mean.1             0.000           0.000
Stddev.1           1.000           1.000
Mean.2             1.299           1.299
Stddev.2           1.000           1.000

$Data.and.Estimates
           Observed Expected Probabilities Residuals  Std.Resid.(analytical)
Rank 1 [1]     2327 2242.671         0.701     1.781                   4.109
Rank 1 [2]      873  957.329         0.299    -2.725                  -4.109
Rank 2 [1]      739  794.802         0.497    -1.979                  -5.318
Rank 2 [2]      351  365.595         0.228    -0.763                  -1.567
Rank 2 [3]      295  248.411         0.155     2.956                   3.582
Rank 2 [4]      215  191.192         0.119     1.722                   3.065
```

Comments:

- The file `MIX-Rank (Ex.2 Multiple forced choice models).R` contains additional code for estimating the data with an extended mixture model as well as SDT restrictions

(corresponding to the unequal variance model). The extended model (for a detailed discussion of this model, see Chapter 3.12.1 and the example in Chapter 15.1.1) exhibits a lack of identification.

- The following files contain examples of the `MIX.Rank` model:

```
MIX-Rank (Ex.0 Parameters & restrictions).R
MIX-Rank (Ex.1 Kellen, Klauer & Singmann, 2012).R
MIX-Rank (Ex.2 Multiple forced choice models).R
```

15.2 MIX.Bias.Rank Example

The following example demonstrates the application of the `MIX.Bias.Rank` model to a data set comprising two sets of 4-AFC data as well as rating data (from an own experiment). The code of this example is contained in the file:

```
MIX-Bias-Rank (Ex.4 Estimate Ranking and rating model (multiple models)).R
```

The relevant code for fitting the data, using extended restrictions, and evaluating as well as printing the results looks like this:

```
datavec <- c(210,  68,  68,  54,     # Ranking 1, Target on Position 1
             193,  98,  60,  49,     # Ranking 1, Target on Position 2
             198, 101,  55,  46,     # Ranking 1, Target on Position 3
             169,  88,  82,  61,     # Ranking 1, Target on Position 4
             180,  79,  78,  63,     # Ranking 2, Target on Position 1
             184,  86,  85,  45,     # Ranking 2, Target on Position 2
             190, 100,  58,  52,     # Ranking 2, Target on Position 3
             185,  86,  74,  55,     # Ranking 2, Target on Position 4
             432, 341, 309, 262, 185,  71,  # Rating data (new)
             179, 220, 238, 256, 257, 450)  # Rating data (old)
cfg <- list(k.rg = c(4, 4), j.rg = c(4, 4), restriction = "EXTENDED")
Opti1.Obj <- SDT.Estimate(data = datavec, n = cfg, Model.Id =
"MIX.Bias.Rank", test = T)
Stat1.Obj <- SDT.Statistics(Opti1.Obj)    # Evaluate the results
print(Stat1.Obj)
```

Here is a portion of the output:

```
$Model.description
[1] "MIX ranking and rating model with bias: Number of Mixture models: <2>
Number of ranking-alternatives: <4, 4> Number of ranking-positions: <4, 4>
Number of quadrature points: <30> Type of restrictions: <extended>"

$Statistics
                                      Statistic
log L                                 -9519.386
X^2                                      28.539
G^2                                      28.428
df                                       20.000
p(Y > X^2)                                0.097
p(Y > G^2)                                0.100
AIC                                   19066.772
BIC                                   19161.469
CAICF                                 19269.956
ICOMP                                 19086.166
ICOMP.R                               19047.139
Free Parameters                          14.000
Length of gradient at optimum             0.000
Rank of Hessian                          14.000
Condition number of information matrix  9018.250
Rank of model matrix: t(J) * J           14.000
Condition number of model matrix      10074.631
```

```
$Free.parameters
                   Value    SE CFI-95(Lower) CFI-95(Upper)
p.mix     (Rank 1)  0.202 0.041         0.122         0.282
Mean.1    (Rank 1)  0.388 0.056         0.278         0.499
Mean.2    (Rank 1)  3.198 1.095         1.052         5.344
Bias.1    (Rank 1)  0.178 0.087         0.006         0.349
Bias.2    (Rank 1)  0.185 0.085         0.019         0.351
Bias.3    (Rank 1)  0.229 0.085         0.063         0.394
Bias.1    (Rank 2) -0.076 0.088        -0.248         0.096
Bias.2    (Rank 2)  0.028 0.086        -0.140         0.195
Bias.3    (Rank 2)  0.076 0.086        -0.093         0.245
c[1]               -0.641 0.029        -0.698        -0.584
c[2]               -0.064 0.026        -0.115        -0.012
c[3]                0.429 0.027         0.376         0.482
c[4]                0.959 0.033         0.894         1.024
c[5]                1.666 0.051         1.567         1.766

$Ranking.parameters
         Rank <k=4, j=4> Rank <k=4, j=4>
p.mix              0.202           0.202
Mean.0             0.000           0.000
Stddev.0           1.000           1.000
Mean.1             0.388           0.388
Stddev.1           1.000           1.000
Mean.2             3.198           3.198
Stddev.2           1.000           1.000
Bias.1             0.178          -0.076
Bias.2             0.185           0.028
Bias.3             0.229           0.076
Bias.4             0.000           0.000

$Rating.parameters
         Gauss-1 Gauss-2
p.mix      1.000   0.202
Mean.0     0.000   0.388
Stddev.0   1.000   1.000
Mean.1     0.000   3.198
Stddev.1   1.000   1.000
t-1       -0.641  -0.641
t-2       -0.064  -0.064
t-3        0.429   0.429
t-4        0.959   0.959
t-5        1.666   1.666
[...]
```

The output is quite similar to that of the `MIX.Rank` model (cf. Chapter 15.1). The main difference consists in the fact that in the section `$Ranking.parameters` listing the parameters of the forced choice models position bias parameters are shown.
The values of the bias parameters indicate that for the first forced choice data set position 4 received less attention than the other 3 positions (these can be seen directly from the data set where the target item was ranked considerably less often as the first item, when it was on the fourth position).
For the second forced choice data set the third position received slightly more weight than the other positions. This is also evident from the frequencies since the target item was ranked more often on the first and second rank when it was on the third position.

The estimated values of the bias parameters are nearly identical to those esteimated by means of the `HTSDT.Bias.Rank` model (cf. Chapter 14.2).

Comments:

- The file `MIX-Bias-Rank (Ex.4 Estimate Ranking and rating model (multiple models)).R` contains additional code for estimating the data with extended mixture model with position bias as well as SDT restrictions (corresponding to the unequal variance model with position bias).
- The following files contain examples of the `MIX.Bias.Rank` model:

```
MIX-Bias-Rank (Ex.0 Parameters & restrictions).R
MIX-Bias-Rank (Ex.1 Estimate Ranking model (1 model only)).R
MIX-Bias-Rank (Ex.3 Estimate Ranking model (multiple models).R
MIX-Bias-Rank (Ex.4 Estimate Ranking and rating model (multiple
models)).R
```

16. Working Example: SDT models with probabilistic item response functions (IRF.Gauss)

In the following, an example of the application of the module `IRF.Gauss` is presented. The example illustrates both probabilistic response model: The graded response model (GRM) and the ordinal Rasch response model (ORM).
The data of Ratcliff, McKoon, & Tindall (1994), Exp. 1, pure strong items are used.
The relevant code for fitting the models to the data, evaluating and printing the results is contained in the file:

```
SDT-IRF-Gauss 1 (2 signals, Data Ratcliff et al. 1994).R
```

The code (including comments) looks like this:

```
# LIBRARY REQUIRED SINCE THE TESTFLAG IN ESTIMATION PROCEDURE IS SET TO TRUE
library(numDeriv)

# LOAD SOURCE FILES WITH MODEL, AUXILIARY AND ESTIMATION FUNCTIONS
source("SDT-IRF-Gauss.R")
source("SDT-Main.R")
source("SDT-Auxiliary.R")

# DATA OF RATCLIFF ET AL. (1994, Exp.1, 200 ms pure strong list)
datavec <- c(477, 776, 527, 321, 258, 184,           # New
             192, 401, 290, 267, 316, 442)           # Old

# FIT SDT-GRM MODEL AND COMPUTE OUTPUT
cfg <- list(n.sdt = 2, model = "GRM")
Opti.GRM <- SDT.Estimate(data = datavec, n = cfg, Model.Id = "IRF.Gauss", test = T)
Stat.GRM <- SDT.Statistics(Opti.GRM)

# FIT SDT-ORM MODEL AND COMPUTE OUTPUT
cfg <- list(n.sdt = 2, model = "ORM")
Opti.ORM <- SDT.Estimate(data = datavec, n = cfg, Model.Id = "IRF.Gauss", test = T)
Stat.ORM <- SDT.Statistics(Opti.ORM)

# PRINT THE RESULTS
cat("\n-------------------------------------------------------------------------\n")
cat(" Results of the GRM-SDT (Graded Response) model:\n")
cat(" Data: Ratcliff, MacKoon, & Tindall, 1994, Exp.1, pure strong items")
cat("\n-------------------------------------------------------------------------\n")
```

```
print(Stat.GRM)
cat("\n------------------------------------------------------------------------\n")
cat(" Results of the ORM-SDT (Ordinal Rasch) model:\n")
cat(" Data: Ratcliff, MacKoon, & Tindall, 1994, Exp.1, pure strong items")
cat("\n------------------------------------------------------------------------\n")
print(Stat.ORM)
```

The code comprises 5 blocks that perform the following actions:

1. The library `NumDeriv` and the source files with the relevant code `SDT-IRF-Gauss.R`, `SDT-Main.R` and `SDT-Auxiliary.R` are loaded.
2. The data vector is `datavec` specified. It contains the counts of the different response categories for new and old items in the order *sure new* to *sure old*.
3. The configuration list `cfg` of the graded response model is specified and the model is fitted using the function `SDT.Estimate()`. The result is stored in the object `Opti.GRM`. Then the function `SDT.Statistics()` computes the relevant output: estimates, statistics etc.
4. The same is done for the ordinal Rasch model.
5. Finally, the results are printed.

The output for the graded response model looks like this:

```
------------------------------------------------------------------------
Results of the GRM-SDT (Graded Response) model:
Data: Ratcliff, MacKoon, & Tindall, 1994, Exp.1, pure strong items
------------------------------------------------------------------------
$Model.description
[1] "Gaussian SDT model with probabilistic item response functions <Model = GRM>,
<Restriction = STANDARD>, <Robust thresholds = FALSE>, <Number of quadrature points
= 35>"

$Statistics
                                       Statistic
log L                                  -7641.877
X^2                                        5.465
G^2                                        5.456
df                                         3.000
p(Y > X^2)                                 0.141
p(Y > G^2)                                 0.141
AIC                                    15297.754
BIC                                    15342.560
CAICF                                  15399.482
ICOMP                                  15288.927
ICOMP.R                                15287.971
Free Parameters                            7.000
Length of gradient at optimum              0.000
Rank of Hessian                            7.000
Condition number of information matrix     44.012
Rank of model matrix: t(J) * J             7.000
Condition number of model matrix           71.835

$Free.parameters
                Value    SE CFI-95(Lower) CFI-95(Upper)
[SDT-2]   Mean  1.214 0.074         1.070         1.358
[SDT-2] Stddev  1.604 0.099         1.409         1.798
c[1]           -1.764 0.057        -1.876        -1.652
c[2]           -0.004 0.046        -0.094         0.086
c[3]            1.006 0.048         0.911         1.100
c[4]            1.834 0.056         1.723         1.944
c[5]            2.941 0.076         2.793         3.090

$Gaussian.parameters
```

```
       Gauss-1 Gauss-2
Mean         0   1.214
Stddev       1   1.604

$Response.parameters
  Difficulty Discrimination
1     -1.764              1
2     -0.004              1
3      1.006              1
4      1.834              1
5      2.941              1

$SDT.measures
       Value    SE CFI-95(Lower) CFI-95(Upper)
d.a[2] 0.908 0.060         0.790         1.026
d.e[2] 0.933 0.059         0.816         1.049
A.z[2] 0.740 0.014         0.713         0.767

$Data.and.Estimates
             Observed Expected Probabilities Residuals Std.Resid.(analytical)
Signal 1 [1]      477  469.449         0.185     0.349                  1.501
Signal 1 [2]      776  800.051         0.315    -0.850                 -1.906
Signal 1 [3]      527  504.819         0.199     0.987                  1.884
Signal 1 [4]      321  322.068         0.127    -0.059                 -0.097
Signal 1 [5]      258  261.165         0.103    -0.196                 -0.301
Signal 1 [6]      184  185.448         0.073    -0.106                 -0.260
Signal 2 [1]      192  199.950         0.105    -0.562                 -1.715
Signal 2 [2]      401  374.938         0.197     1.346                  2.221
Signal 2 [3]      290  311.196         0.163    -1.202                 -2.002
Signal 2 [4]      267  267.650         0.140    -0.040                 -0.067
Signal 2 [5]      316  313.933         0.165     0.117                  0.218
Signal 2 [6]      442  440.334         0.231     0.079                  0.436
```

The output for the ordinal Rasch model looks like this:

```
------------------------------------------------------------------------
 Results of the ORM-SDT (Ordinal Rasch) model:
 Data: Ratcliff, MacKoon, & Tindall, 1994, Exp.1, pure strong items
------------------------------------------------------------------------
$Model.description
[1] "Gaussian SDT model with probabilistic item response functions <Model = ORM>,
<Restriction = STANDARD>, <Robust thresholds = FALSE>, <Number of quadrature points
= 35>"

$Statistics
                                       Statistic
log L                                  -7641.052
X^2                                        3.814
G^2                                        3.807
df                                         3.000
p(Y > X^2)                                 0.282
p(Y > G^2)                                 0.283
AIC                                    15296.105
BIC                                    15340.911
CAICF                                  15395.003
ICOMP                                  15283.680
ICOMP.R                                15283.335
Free Parameters                            7.000
Length of gradient at optimum              0.000
Rank of Hessian                            7.000
Condition number of information matrix      8.458
Rank of model matrix: t(J) * J             7.000
Condition number of model matrix          18.550

$Free.parameters
               Value    SE CFI-95(Lower) CFI-95(Upper)
[SDT-2]   Mean  0.802 0.049         0.705         0.898
```

```
[SDT-2] Stddev  1.313 0.066         1.184         1.443
c[1]           -1.190 0.060        -1.308        -1.072
c[2]            0.311 0.054         0.206         0.416
c[3]            0.792 0.061         0.673         0.912
c[4]            1.006 0.068         0.873         1.140
c[5]            1.504 0.082         1.342         1.666

$Gaussian.parameters
       Gauss-1 Gauss-2
Mean         0   0.802
Stddev       1   1.313

$Response.parameters
  Difficulty Discrimination
1     -1.190              1
2      0.311              1
3      0.792              1
4      1.006              1
5      1.504              1

$SDT.measures
       Value    SE CFI-95(Lower) CFI-95(Upper)
d.a[2] 0.687 0.043         0.602         0.771
d.e[2] 0.693 0.043         0.609         0.777
A.z[2] 0.686 0.011         0.665         0.708

$Data.and.Estimates
             Observed Expected Probabilities Residuals  Std.Resid.(analytical)
Signal 1 [1]      477  468.788         0.184     0.379                   1.418
Signal 1 [2]      776  797.808         0.314    -0.772                  -1.654
Signal 1 [3]      527  511.768         0.201     0.673                   1.290
Signal 1 [4]      321  323.154         0.127    -0.120                  -0.199
Signal 1 [5]      258  254.025         0.100     0.249                   0.378
Signal 1 [6]      184  187.457         0.074    -0.253                  -0.646
Signal 2 [1]      192  200.280         0.105    -0.585                  -1.629
Signal 2 [2]      401  378.782         0.199     1.142                   1.904
Signal 2 [3]      290  304.964         0.160    -0.857                  -1.450
Signal 2 [4]      267  264.931         0.139     0.127                   0.220
Signal 2 [5]      316  320.353         0.168    -0.243                  -0.470
Signal 2 [6]      442  438.690         0.230     0.158                   0.749
```

For a detailed description of the output, cf. Chapter 5.1 (page 105).

16.1 List of example files forGaussian SDT models with probabilistic response functions

```
SDT-IRF-Gauss 1 (2 signals, Data Ratcliff et al. 1994).R
SDT-IRF-Gauss 2 (3 signals, Data Ratcliff et al. 1994).R
SDT-IRF-Gauss 3 (Y-N-Data).R
SDT-IRF-Gauss 4 (Parameterinfo).R
```

17. References

Agresti, A. (2002*). Categorical data analysis* (2nd edition). New York: Wiley.

Arndt, J. & Reder, L. M. (2002). Word frequency and receiver operating characteristic curves in recognition memory: Evidence for a dual-process interpretation. *Journal of Experimental Psychology: Learning, Memory, & Cognition, 28*, 830-842.

Ashby, F. G., & Townsend, J. T. (1986). Varieties of perceptual independence. *Psychological Review, 93*, 154-179.

Bozdogan, H. (1987). Model selection and Akaike's information criterion (AIC): The general theory and its analytic extensions. *Psychometrika, 52*, 345-370.

Bozdogan, H. (1988). ICOMP: A new model selection criterion. In: H. H. Bock (Ed.), *Classification and related methods of data analysis* (pp. 599-608). Amsterdam: North-Holland.

DeCarlo, L. T. (2003a). An application of signal detection theory with finite mixture distributions of source discrimination. *Journal of Experimental Psychology, Learning, Memory, & Cognition, 29,* 767-778.

DeCarlo, L. T. (2003b). Source monitoring and multivariate signal detection theory, with a model for selection. *Journal of Mathematical Psychology, 47,* 292-303.

DeCarlo, L. T. (2007). The mirror effect and mixture signal detection theory. *Journal of Experimental Psychology, Learning, Memory, & Cognition, 33,* 18-33.

DeCarlo, L. T. (2008). Process dissociation and mixture signal detection theory. *Journal of Experimental Psychology, Learning, Memory, & Cognition, 34,* 1565-1572.

DeCarlo, L. T. (2012). On a signal detection approach to m-alternative forced choice with bias, with maximum likelihood and Bayesian approaches to estimation. *Journal of Mathematical Psychology, 56,* 196-207.

Donaldson, W., & Good, C. (1996). A'r: An estimates of area under isosensitivity curves. *Behavior Research Methods, Instruments, & Computers, 28,* 590-597.

Ennis, D. M., & O'Mahony, M. (1995). Probabilistic models of sequential taste effects in triadic choice. *Journal of Experiment Psychology: Human Perception and Performance, 21,* 1088-1097.

Gilbert, P. (2006). *numDeriv: Accurate Numerical Derivatives. R package version 2006.4-1.*

Greene, W. H. (2008). *Econometric analysis* (6th edition). Upper Saddle River: Prentice Hall.

Hanley, J. A., & McNeil, R.J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology, 143,* 29-36.

Hilford, A., Glanzer, M., Kim, K., & DeCarlo, L. T. (2002). Regularities of source recognition: ROC analysis. *Journal of Experimental Psychology: General, 131,* 494-510

Kellen, D., Klauer K. C., & Singmann, H. (2012). On the measurement of criterion noise in signal detection theory: The case of recognition memory. *Psychological Review, 119,* 457-479.

Kelley, R., & Wixted, J. T. (2001). On the nature of associative information in recognition memory. *Journal of Experimental Psychology: Learning, Memory, & Cognition, 27,* 701-722.

Macho, S. (2002). Cognitive modeling with spreadsheets. *Behavior Research Methods, Instruments, & Computers, 34,* 19-36.

Macho, S. (2004). Modeling associative recognition: A comparison of two-high-threshold, two-high-threshold signal detection, and mixture distribution models. *Journal of Experimental Psychology: Learning, Memory, & Cognition, 30,* 83-97.

Macmillan, N. A., & Creelman, C. D. (2005). *Detection theory: A user's guide* (2nd edition). Cambridge: Cambridge University Press.

McDonald, R. P., & Krane, W. R. (1979). A Monte Carlo study of local identifiability and degrees of freedom in the asymptotic likelihood ratio test. *British Journal of Mathematical Psychology, 32*, 121-132.

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika, 47*, 149-174. doi: 10.1007/BF02296272

Masters, G. N. (2010). The partial credit model. In M. L. Nering, & R. Ostini, (Eds.), *Handbook of polytomous item response models* (pp. 109-122). New York: Taylor & Francis.

Masters, G. N., & Wright, B. D. (1984). The essential process in a family of measurement models. *Psychometrika, 49,* 529-544.

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement, 16*, 159-176. doi: 10.1177/014662169201600206

Olzak, L. A., & Kramer, P. (1984). Interactions between spatially tuned mechanisms: Converging evidence. *Journal of the Optical Society of America A, 1,* 1290 (Abstract).

Onyper, S. V., Zhang, Y. X., & Howard, M. W. (2010). Some-or-none recollection: Evidence from item and source memory. *Journal of Experimental Psychology: General, 139,* 341-364.

Pawitan, Y. (2001). *In all likelihood: Statistical modelling and inference using likelihood*. Oxford; UK: Clarendon Press.

Rao, C. P. (1973). *Linear statistical inference and its applications* (2nd edition). New York: Wiley.

Ratcliff, R., & McKoon, G., & Tindall, M. (1994). Empirical generality of data from recognition memory receiver-operating characteristic functions and implications for global memory models. *Journal of Experimental Psychology: Learning, Memory, & Cognition, 20,* 763-785.

Rotello, C. M., Macmillan, N. A., & Reeder, J. A. (2004). Sum-difference theory of remembering and knowing: A two-dimensional signal-detection model. *Psychological Review, 111*, 588-616.

Rotello, C. M., Macmillan, N. A., Reeder, J. A., & Wong, M. (2005). The remember response: Subject to bias, graded, and not a process-pure indicator of recollection. *Psychonomic Bulletin & Review, 15,* 865-873.

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika, Monograph Supplement No.17.*

Samejima, F. (1997). Graded response model. In W. J. Van der Linden & R. K. Hambleton, (Eds.), *Handbook of modern item response theory* (pp.85-100). New York: Springer.

Samejima, F. (2010). The general graded response model. In M. L. Nering, & R. Ostini, (Eds.), *Handbook of polytomous item response models* (pp. 77-107). New York: Taylor & Francis.

Simpson, A. J., & Fitter, M. J. (1973). What is the best index of detectability? *Psychological Bulletin, 80,* 481-488.

Thomas, R. D. (2001). Characterizing perceptual interactions in face identification using multidimensional signal detection theory. In M. J. Wenger, & J. T. Townsend (Eds.), *Computational geometry, and processing perspectives on facial cognition* (Chapter 6, pp. 193-227). Hillsdale, NJ: Erlbaum.

Vokey, J. R. (2016). Single-step simple ROC curve fitting via PCA. *Canadian Journal of Experimental Psychology, 70,* 301-305. doi: 10.1037/cep0000095

Wickens, T. D. (1992). Maximum-Likelihood estimation of a multivariate Gaussian rating model with excluded data. *Journal of Mathematical Psychology, 36,* 213-234.

Wickens, T. D. (2002). *Elementary signal detection theory.* Oxford: Oxford University Press

Wickens, T. D., & Olzak, L. A. (1992). Three views of association in concurrent detection ratings. In: F. G. Ashby (Ed.), *Multidimensional models of perception and cognition* (Chapter 9: pp. 229-252). Hillsdale, NJ.: Erlbaum.

Wixted, J. T., & Mikes, L. (2010). A continuous dual-process model of remember/know judgments. *Psychological Review, 117,* 1025-1054.

Yonelinas, A. P. (1994). Receiver-operating characteristics in recognition memory: Evidence for a dual-process model. *Journal of Experimental Psychology: Learning, Memory, & Cognition, 20,* 1341-1354.

Yonelinas, A. P. (1999). The contribution of recollection and familiarity to recognition and source-memory judgments: A formal dual-process model and an analysis of receiver operating characteristics. *Journal of Experimental Psychology: Learning, Memory, & Cognition, 25,* 1415-1434.