

# Package: mutoss (via r-universe)

October 31, 2024

**Type** Package

**Title** Unified Multiple Testing Procedures

**Version** 0.1-13

**Author** MuToss Coding Team (Berlin 2010), Gilles Blanchard, Thorsten Dickhaus, Niklas Hack, Frank Konietzke, Kornelius Rohmeyer, Jonathan Rosenblatt, Marsel Scheer, Wiebke Werft

**Maintainer** Kornelius Rohmeyer <rohmeier@small-projects.de>

**Description** Designed to ease the application and comparison of multiple hypothesis testing procedures for FWER, gFWER, FDR and FDX. Methods are standardized and usable by the accompanying 'mutossGUI'.

**Depends** R (>= 2.10.0), mvtnorm

**Suggests** fdrtool, qvalue, testthat, lattice

**Imports** plotrix, multtest (>= 2.2.0), multcomp (>= 1.1-0), methods

**License** GPL

**LazyLoad** yes

**LazyData** true

**URL** <https://github.com/kornl/mutoss/>

**BugReports** <https://github.com/kornl/mutoss/issues/>

**Repository** <https://kornl.r-universe.dev>

**RemoteUrl** <https://github.com/kornl/mutoss>

**RemoteRef** HEAD

**RemoteSha** d72dacd510386df4b53fcfe94121abac23a16b29

## Contents

|                       |   |
|-----------------------|---|
| ABH_pi0_est . . . . . | 3 |
| adaptiveBH . . . . .  | 4 |
| adaptiveSTS . . . . . | 5 |

|                         |    |
|-------------------------|----|
| adjPValuesPlot          | 6  |
| aorc                    | 7  |
| augmentation            | 8  |
| BH                      | 9  |
| BL                      | 11 |
| BlaRoq                  | 12 |
| bonferroni              | 13 |
| BR_pi0_est              | 14 |
| BY                      | 15 |
| calculateBetaAdjustment | 16 |
| compareMutoss           | 17 |
| ErrorControl-class      | 19 |
| fisher22.marginal       | 19 |
| fisher22_fast           | 20 |
| fisher23.marginal       | 21 |
| fisher23_fast           | 21 |
| ftest.marginal          | 22 |
| gao                     | 23 |
| gatherParameters        | 24 |
| gatherStatistics        | 25 |
| hochberg                | 27 |
| holm                    | 28 |
| hommel                  | 29 |
| indepBR                 | 30 |
| jointCDF.orderedUnif    | 31 |
| linearStepUp            | 32 |
| mu.test.class           | 33 |
| mu.test.name            | 34 |
| mu.test.rates           | 34 |
| mu.test.same.data       | 35 |
| mu.test.type            | 36 |
| multcomp.wrapper        | 36 |
| multiple.down           | 38 |
| multiple.down.adjust    | 39 |
| Mutoss-class            | 39 |
| mutoss.apply            | 40 |
| mutoss.models           | 41 |
| mutoss.plotCI           | 42 |
| MutossMethod-class      | 42 |
| notterman               | 43 |
| nparcomp                | 43 |
| nparcomp.wrapper        | 45 |
| onesamp.marginal        | 46 |
| oracleBH                | 47 |
| paired.marginal         | 48 |
| printRejected           | 49 |
| pval2locfdr             | 49 |
| pval2qval               | 50 |

|                            |           |
|----------------------------|-----------|
| pValuesPlot . . . . .      | 51        |
| Qvalue . . . . .           | 52        |
| ranktruncated . . . . .    | 53        |
| regwq . . . . .            | 54        |
| reject . . . . .           | 55        |
| requireLibrary . . . . .   | 56        |
| rom . . . . .              | 56        |
| SD . . . . .               | 57        |
| sidak . . . . .            | 58        |
| SidakSD . . . . .          | 59        |
| simulation . . . . .       | 60        |
| snk . . . . .              | 62        |
| storey_pi0_est . . . . .   | 64        |
| SU . . . . .               | 65        |
| SUD . . . . .              | 65        |
| TSBKY_pi0_est . . . . .    | 66        |
| tukey.wrapper . . . . .    | 67        |
| two.stage . . . . .        | 68        |
| twosamp.marginal . . . . . | 69        |
| twostageBR . . . . .       | 70        |
| <b>Index</b>               | <b>72</b> |

---

|             |  |
|-------------|--|
| ABH_pi0_est | <i>Lowest Slope Line (LSL) method of Hochberg and Benjamini for estimating pi0</i> |
|-------------|--|

---

## Description

The Lowest Slope Line (LSL) method of Hochberg and Benjamini for estimating  $\pi_0$  is applied to p-values. This method for estimating  $\pi_0$  is motivated by the graphical approach proposed by Schweder and Spjøtvoll (1982), as developed and presented in Hochberg and Benjamini (1990).

## Usage

```
ABH_pi0_est(pValues)
```

## Arguments

|         |   |
|---------|---|
| pValues | The raw p-values for the marginal test problems |
|---------|---|

## Value

|              |   |
|--------------|---|
| $\pi_0$ .ABH | The estimated proportion of true null hypotheses. |
|--------------|---|

## Author(s)

WerftWiebke

## References

- Hochberg, Y. and Benjamini, Y. (1990). More powerful procedures for multiple significance testing. *Statistics in Medicine* 9, 811-818.
- Schweder, T. and Spjøtvoll, E. (1982). Plots of P-values to evaluate many tests simultaneously. *Biometrika* 69, 3, 493-502.

## Examples

```
my.pvals <- c(runif(50), runif(50, 0, 0.01))
result <- ABH_pi0_est(my.pvals)
```

---

 adaptiveBH

*Benjamini-Hochberg (2000) adaptive linear step-up procedure*


---

## Description

The adaptive Benjamini-Hochberg step-up procedure is applied to pValues. It controls the FDR at level alpha for independent or positive regression dependent test statistics.

## Usage

```
adaptiveBH(pValues, alpha, silent=FALSE)
```

## Arguments

|         |   |
|---------|---|
| pValues | The used raw pValues.                                 |
| alpha   | The level at which the FDR shall be controlled.       |
| silent  | If true any output on the console will be suppressed. |

## Details

In the adaptive Benjamini-Hochberg step-up procedure the number of true null hypotheses is estimated first as in Hochberg and Benjamini (1990), and this estimate is used in the procedure of Benjamini and Hochberg (1995) with  $\alpha' = \alpha * m/m_0$ . Please note that this method is not equivalent to multcomp's ABH. They revised the formular of the original paper.

## Value

A list containing:

|                |  |
|----------------|--|
| adjPValues     | A numeric vector containing the adjusted pValues   |
| criticalValues | A numeric vector containing critical values used in the step-up-down test  |
| rejected       | A logical vector indicating which hypotheses are rejected  |
| pi0            | An estimate of the proportion of true null hypotheses among all hypotheses ( $\pi_0 = m_0/m$ ).                      |
| errorControl   | A Mutoss S4 class of type errorControl, containing the type of error controlled by the function and the level alpha. |

**Author(s)**

WerftWiebke

**References**

Benjamini, Y. and Hochberg, Y. (2000). On the Adaptive Control of the False Discovery Rate in Multiple Testing With Independent Statistics. *Journal of Educational and Behavioral Statistics*, 25(1): 60-83.*n*

Hochberg, Y. and Benjamini, Y. (1990). More powerful procedures for multiple significance testing. *Statistics in Medicine* 9, 811-818.*n*

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B*, 57:289-300.

**Examples**

```
alpha <- 0.05
p <-c(runif(10, min=0, max=0.01), runif(10, min=0.9, max=1))
result <- adaptiveBH(p, alpha)
result <- adaptiveBH(p, alpha, silent=TRUE)
```

---

adaptiveSTS

---

*Storey-Taylor-Siegmund (2004) adaptive step-up procedure*


---

**Description**

Storey-Taylor-Siegmund's (2004) adaptive step-up procedure

**Usage**

```
adaptiveSTS(pValues, alpha, lambda=0.5, silent=FALSE)
```

**Arguments**

|         |   |
|---------|---|
| pValues | The used raw pValues.   |
| alpha   | The level at which the FDR shall be controlled.                     |
| lambda  | The tuning parameter for the estimation procedure (defaults to 0.5) |
| silent  | If true any output on the console will be suppressed.               |

**Details**

The adaptive STS procedure uses a conservative estimate of  $\pi_0$  which is plugged in a linear step-up procedure. The estimation of  $\pi_0$  requires a parameter ( $\lambda$ ) which is set to 0.5 by default. Note that the estimated  $\pi_0$  is truncated at 1 as suggested by the author, so the implementation of the procedure is not entirely supported by the proof in the reference.

**Value**

A list containing:

adjPValues     A numeric vector containing the adjusted pValues  
rejected        A logical vector indicating which hypotheses are rejected  
criticalValues  A numeric vector containing critical values used in the step-up-down test  
errorControl    A Mutoss S4 class of type errorControl, containing the type of error controlled by the function and the level alpha.

**Author(s)**

Werft Wiebke

**References**

Storey, J.D., Taylor, J.E. and Siegmund, D. (2004). Strong control, conservative point estimation and simultaneous conservative consistency of false discovery rates: a unified approach. *Journal of the Royal Statistical Society, B* 66(1):187-205.

**Examples**

```
alpha <- 0.05
p <-c(runif(10, min=0, max=0.01), runif(10, min=0.9,max=1))
result <- adaptiveSTS(p, alpha, lambda=0.5)
result <- adaptiveSTS(p, alpha, lambda=0.5, silent=TRUE)
```

---

adjPValuesPlot     *A function plotting adjusted p-values...*

---

**Description**

A function plotting adjusted p-values

**Usage**

```
adjPValuesPlot(adjPValues, alpha)
```

**Arguments**

adjPValues     A numeric containing the adjusted pValues to plot.  
alpha          A numeric containing the overall type I error rate alpha..

**Author(s)**

MuToss-Coding Team

---

aorc

*Step-up-down test based on the asymptotically optimal rejection curve.*


---

### Description

Performs a step-up-down test on pValues. The critical values are based on the asymptotically optimal rejection curve. To have finite FDR control, an automatic adjustment of the critical values is done (see details for more).

### Usage

```
aorc(pValues, alpha, startIDX_SUD=length(pValues), betaAdjustment,
     silent=FALSE)
```

### Arguments

|                |   |
|----------------|---|
| pValues        | pValues to be used. They are assumed to be stochastically independent.  |
| alpha          | The level at which the FDR shall be controlled.   |
| startIDX_SUD   | The index at which critical value the step-up-down test starts. Default is length(pValues) and thus the function aorc() by default behaves like an step-up test.                              |
| betaAdjustment | A numeric value to adjust the asymptotically optimal rejection curve for the finite sample case. If betaAdjustment is not set an algorithm will calculate it, but this can be time-consuming. |
| silent         | If true any output on the console will be suppressed.   |

### Details

The graph of the function  $f(t) = t / (t * (1 - \alpha) + \alpha)$  is called the asymptotically optimal rejection curve. Denote by  $\text{finv}(t)$  the inverse of  $f(t)$ . Using the critical values  $\text{finv}(i/n)$  for  $i = 1, \dots, n$  yields asymptotic FDR control. To ensure finite control it is possible to adjust  $f(t)$  by a factor. The function `calculateBetaAdjustment()` calculates a beta such that  $(1 + \text{beta} / n) * f(t)$  can be used to control the FDR for a given finite sample size. If the parameter `betaAdjustment` is not provided, `calculateBetaAdjustment()` will be called automatically.

### Value

A list containing:

|                |  |
|----------------|--|
| adjPValues     | A numeric vector containing the adjusted pValues   |
| rejected       | A logical vector indicating which hypotheses are rejected  |
| criticalValues | A numeric vector containing critical values used in the step-up-down test  |
| errorControl   | A Mutoss S4 class of type <code>errorControl</code> , containing the type of error (FDR) controlled by the function and the level alpha. |

**Author(s)**

MarselScheer

**References**

Finner, H., Dickhaus, T. & Roters, M. (2009). On the false discovery rate and an asymptotically optimal rejection curve. *The Annals of Statistics* 37, 596-618.

**Examples**

```
## Not run: # Takes more than 6 seconds therefor CRAN should not check it:
r <- c(runif(10), runif(10, 0, 0.01))
result <- aorc(r, 0.05)
result <- aorc(r, 0.05, startIDX_SUD = 1) ## step-down
result <- aorc(r, 0.05, startIDX_SUD = length(r)) ## step-up
## End(Not run)
```

---

|              |  |
|--------------|--|
| augmentation | <i>Wrapper function to the augmentation methods of the multtest package.</i> |
|--------------|--|

---

**Description**

Wrapper function to the augmentation methods of the multtest package.

**Usage**

```
augmentation(adjPValues, newErrorControl, newK, newQ, silent=FALSE)
```

**Arguments**

|                 |  |
|-----------------|--|
| adjPValues      | a vector of p-values that are <i>*already*</i> adjusted for FWER     |
| newErrorControl | new error control type to adjust for. One of ("gFWER", "FDX", "FDR") |
| newK            | k-parameter if newErrorControl is "gFWER"                            |
| newQ            | q-parameter if newErrorControl is "FDX"                              |
| silent          | if true any output on the console will be suppressed.                |

**Details**

The augmentation method turns a vector of p-values which are already adjusted for FWER control into p-values that are adjusted for gFWER, FDX or FDR. The underlying idea (for gFWER and FDX) is that the set of hypotheses rejected at a given level alpha under FWER can be "augmented" by rejecting some additional hypotheses while still ensuring (strong) control of the desired weaker type I criterion. For FDR, it uses the fact that FDX control for  $q=\alpha=1-\sqrt{1-\beta}$  entails FDR control at level beta.



Use of these augmentation methods is recommended only in the situation where FWER-controlled p-values are directly available from the data (using some specific method). When only marginal p-values are available, it is generally preferable to use other adjustment methods directly aimed at the intended criterion (as opposed to first adjust for FWER, then augment)

Note: in the multtest package, two methods ("restricted" and "conservative") are available for FDR augmentation. Here the 'restricted' method is forced for FDR augmentation since it is in fact always valid and better than "conservative" (M. van der Laan, personal communication) with respect to power.

### Value

A list containing:

|              |  |
|--------------|--|
| adjPValues   | A numeric vector containing the new adjusted pValues   |
| rejected     | A logical vector indicating which hypotheses are rejected. Currently always NULL.                |
| errorControl | A Mutoss S4 class of type errorControl, containing the type of error controlled by the function. |

### Author(s)

KornRohm, GillesBlanchard

### References

Dudoit, S. and van der Laan, M.J. (2008) Multiple Testing Procedures with Applications to Genomics, Springer. (chapter 6)

### Examples

```
## Not run:  TODO NH (MS) EXAMPLE PROBLEM
p <- c(runif(50), runif(50, 0, 0.01))
fwer_adj <- bonferroni(p) # adjust for FWER using Bonferroni correction
fdx_adj <- augmentation(fwer_adj$adjPValues, "FDX", q=0.1, silent = TRUE) #augment to FDX (q=0.1)

## End(Not run)
```

---

BH

*Benjamini-Hochberg (1995) linear step-up procedure*

---

### Description

Benjamini-Hochbergs Linear Step-Up Procedure. The procedure controls the FDR when the test statistics are stochastically independent or satisfy positive regression dependency (PRDS) (see Benjamini and Yekutieli 2001 for details). The Benjamini-Hochberg (BH) step-up procedure considers ordered pValues  $P_{(i)}$ . It defines  $k$  as the largest  $i$  for which  $P_{(i)} \leq i \cdot \alpha / m$  and then rejects all associated hypotheses  $H_{(i)}$  for  $i=1, \dots, k$ . In their seminal paper, Benjamini and Hochberg (1995)

show that for  $0 \leq m_0 \leq m$  independent pValues corresponding to true null hypotheses and for any joint distribution of the  $m_1 = m - m_0$  p-values corresponding to the non null hypotheses, the FDR is controlled at level  $(m_0/m) \cdot \alpha$ . Under the assumption of the PRDS property, (for details see Benjamini and Yekutieli 2001). In Benjamini et al. (2006) the BH procedure is improved by adaptive procedures which use an estimate of  $m_0$  and apply the BH method at level  $\alpha' = \alpha \cdot m/m_0$ , to fully exhaust the desired level  $\alpha$  (see Adaptive Benjamini Hochberg and Two Stage Benjamini Yekutieli).

### Usage

```
BH(pValues, alpha, silent=FALSE)
```

### Arguments

|         |   |
|---------|---|
| pValues | The used unadjusted pValues.                          |
| alpha   | The level at which the FDR shall be controlled.       |
| silent  | If true any output on the console will be suppressed. |

### Value

A list containing:

|                |  |
|----------------|--|
| adjPValues     | A numeric vector containing the adjusted pValues   |
| criticalValues | A numeric vector containing critical values used in the step-up test   |
| rejected       | A logical vector indicating which hypotheses are rejected  |
| errorControl   | A Mutoss S4 class of type errorControl, containing the type of error controlled by the function and the level alpha. |

### Author(s)

Werft Wiebke

### References

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B*, 57:289-300.*n*

Benjamini, Y. and Yekutieli, D. (2001). The control of the false discovery rate in multiple testing under dependency. *Annals of Statistics*, 29(4):1165-1188.*n*

Benjamini, Y., Krieger, A. and Yekutieli, D. (2006). Adaptive linear step-up procedures that control the false discovery rate. *Biometrika*, 93(3):491-507.

### Examples

```
alpha <- 0.05
p <- c(runif(10, min=0, max=0.01), runif(10, min=0.9, max=1))
result <- BH(p, alpha)
result <- BH(p, alpha, silent=TRUE)
```

---

BL *Benjamini-Liu (1999) step-down procedure*

---

### Description

Benjamini-Liu's step-down procedure is applied to pValues. The procedure controls the FDR if the corresponding test statistics are stochastically independent.

### Usage

```
BL(pValues, alpha, silent=FALSE)
```

### Arguments

|         |   |
|---------|---|
| pValues | Numeric vector of p-values                            |
| alpha   | The level at which the FDR is to be controlled.       |
| silent  | If true any output on the console will be suppressed. |

### Details

The Benjamini-Liu (BL) step-down procedure neither dominates nor is dominated by the Benjamini-Hochberg (BH) step-up procedure. However, in Benjamini and Liu (1999) a large simulation study concerning the power of the two procedures reveals that the BL step-down procedure is more suitable when the number of hypotheses is small. Moreover, if most hypotheses are far from the null then the BL step-down procedure is more powerful than the BH step-up method. The BL step-down method calculates critical values according to Benjamin and Liu (1999), i.e.,  $c_i = 1 - (1 - \min(1, (m \cdot \alpha) / (m - i + 1)))^{1/(m - i + 1)}$  for  $i = 1, \dots, m$ , where  $m$  is the number of hypotheses tested. Then, let  $k$  be the smallest  $i$  for which  $P_{(i)} > c_i$  and reject the associated hypotheses  $H_{(1)}, \dots, H_{(k-1)}$ .

### Value

A list containing:

|                |  |
|----------------|--|
| adjPValues     | A numeric vector containing the adjusted pValues.  |
| criticalValues | A numeric vector containing critical values used in the step-up-down test.   |
| rejected       | A logical vector indicating which hypotheses are rejected.   |
| errorControl   | A Mutoss S4 class of type errorControl, containing the type of error controlled by the function and the level alpha. |

### Author(s)

Werft Wiebke

### References

Benjamini, Y. and Liu, W. (1999). A step-down multiple hypotheses testing procedure that controls the false discovery rate under independence. *Journal of Statistical Planning and Inference* Vol. 82(1-2): 163-170.

**Examples**

```
alpha <- 0.05
p <-c(runif(10, min=0, max=0.01), runif(10, min=0.9, max=1))
result <- BL(p, alpha)
result <- BL(p, alpha, silent=TRUE)
```

---

|       |   |
|-------|---|
| BlaRq | <i>Blanchard-Roquain (2008) step-up Procedure for arbitrary dependent p-Values...</i> |
|-------|---|

---

**Description**

Blanchard-Roquain (2008) step-up Procedure for arbitrary dependent p-Values Also proposed independently by Sarkar (2008)

**Usage**

```
BlaRq(pValues, alpha, pii, silent=FALSE)
```

**Arguments**

|         |  |
|---------|--|
| pValues | pValues to be used. They can have arbitrary dependence.                |
| alpha   | the level at which the FDR should be controlled                        |
| pii     | Prior for the proportion of true null hypotheses, same size as pValues |
| silent  | if true any output on the console will be suppressed.                  |

**Details**

A generalization of the Benjamini-Yekutieli procedure, taking as an additional parameter a distribution  $\text{pii}$  on  $[1..k]$  ( $k$  is the number of hypotheses) representing prior belief on the number of hypotheses that will be rejected.

It is a step-up Procedure with critical values  $C_i$  defined as  $\alpha/k$  times the sum for  $j$  in  $[1..i]$  of  $j \cdot \text{pii}[j]$ . For any fixed prior  $\text{pii}$ , the FDR is controlled at level  $\alpha$  for arbitrary dependence structure of the p-Values. The particular case of the Benjamini-Yekutieli step-up is recovered by taking  $\text{pii}[i]$  proportional to  $1/i$ .

If  $\text{pii}$  is missing, a default prior distribution proportional to  $\exp(-i/(0.15 \cdot k))$  is taken. It should perform better than the BY procedure if more than about 0.05 to 0.1 of hypotheses are rejected, and worse otherwise.

Note: the procedure automatically normalizes the prior  $\text{pii}$  to sum to one if this is not the case.

**Value**

A list containing:

|                |  |
|----------------|--|
| adjPValues     | A numeric vector containing the adjusted pValues   |
| rejected       | A logical vector indicating which hypotheses are rejected  |
| criticalValues | A numeric vector containing critical values used in the step-up test   |
| errorControl   | A Mutoss S4 class of type errorControl, containing the type of error controlled by the function and the level alpha. |

**Author(s)**

GillesBlanchard,HackNiklas

**References**

Blanchard, G. and Roquain, E. (2008). Two simple sufficient conditions for FDR control. *Electronic Journal of Statistics*, 2:963-992. Sarkar, S.K. (2008) On methods controlling the false discovery rate. *Sankhya, Series A*, 70:135-168.

---

|            |                                 |
|------------|---------------------------------|
| bonferroni | <i>Bonferroni correction...</i> |
|------------|---------------------------------|

---

**Description**

Bonferroni correction

**Usage**

```
bonferroni(pValues, alpha, silent=FALSE)
```

**Arguments**

|         |  |
|---------|--|
| pValues | A numeric vector containing the unadjusted pValues. No assumption is made on the dependence structure. |
| alpha   | The overall type I error at which the FWER shall be controlled (optional).                             |
| silent  | logical scalar. If TRUE no output is generated.  |

**Details**

The classical Bonferroni correction outputs adjusted p-values, ensuring strong FWER control under arbitrary dependence of the input p-values. It simply multiplies each input p-value by the total number of hypotheses (and ceils at value 1).

It is recommended to use Holm's step-down instead, which is valid under the exact same assumptions and more powerful.

**Value**

A list containing:

|              |  |
|--------------|--|
| adjPValues   | A numeric vector containing the new adjusted pValues   |
| rejected     | (if alpha is given) A logical vector indicating which hypotheses are rejected                    |
| errorControl | A Mutoss S4 class of type errorControl, containing the type of error controlled by the function. |

**Author(s)**

MuToss-Coding Team

**References**

- Bonferroni, C. E. (1935) Il calcolo delle assicurazioni su gruppi di teste. 'In Studi in Onore del Professore Salvatore Ortu Carboni. Rome: Italy, pp. 13-60.
- Bonferroni, C. E. (1936) Teoria statistica delle classi e calcolo delle probabilita. Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze 8, 3-62, 1936.

---

BR\_pi0\_est

*Estimate of pi0 using the one-step Blanchard-Roquain procedure*


---

**Description**

The proportion of true nulls is estimated using the Blanchard-Roquain 1-stage procedure with parameter (alpha,lambda) via the formula

**Usage**

```
BR_pi0_est(pValues, alpha, lambda=1, truncate=TRUE)
```

**Arguments**

|          |   |
|----------|---|
| pValues  | The raw p-values for the marginal test problems (assumed to be independent)   |
| alpha    | The FDR significance level for the BR procedure                               |
| lambda   | (default 1) The parameter for the BR procedure, should belong to (0, 1/alpha) |
| truncate | (logical, default TRUE) if TRUE, output estimated is truncated to 1           |

**Details**

$$\text{estimated pi}_0 = (m - R(\alpha, \lambda) + 1) / (m * (1 - \lambda * \alpha))$$

where  $R(\alpha, \lambda)$  is the number of hypotheses rejected by the BR 1-stage procedure,  $\alpha$  is the FDR level for this procedure and  $\lambda$  a parameter belonging to  $(0, 1/\alpha)$  with default value 1. Independence of p-values is assumed. This estimate may in some cases be larger than 1; it is truncated to 1 if the parameter truncated=TRUE. The estimate is used in the Blanchard-Roquain 2-stage step-up (using the non-truncated version)

**Value**

`pi0` The estimated proportion of true null hypotheses.

**Author(s)**

GillesBlanchard

**References**

Blanchard

BY *Benjamini-Yekutieli (2001) step-up procedure*

**Description**

The Benjamini-Yekutieli step-up procedure is applied to pValues. The procedure ensures FDR control for any dependency structure.

**Usage**

```
BY(pValues, alpha, silent=FALSE)
```

**Arguments**

`pValues` The used unadjusted pValues.  
`alpha` The level at which the FDR shall be controlled.  
`silent` If true any output on the console will be suppressed.

**Details**

The critical values of the Benjamini-Yekutieli (BY) procedure are calculated by replacing the alpha of the Benjamini-Hochberg procedure by  $\alpha/\sum(1/1:m)$ , i.e.,  $c(i)=i*\alpha/(m*(\sum(1/1:m)))$  for  $i=1,\dots,m$ . For large number  $m$  of hypotheses the critical values of the BY procedure and the BH procedure differ by a factor  $\log(m)$ . Benjamini and Yekutieli (2001) showed that this step-up procedure controls the FDR at level  $\alpha*m/m_0$  for any dependency structure among the test statistics.

**Value**

A list containing:

`adjPValues` A numeric vector containing the adjusted pValues  
`criticalValues` A numeric vector containing critical values used in the step-up-down test  
`rejected` A logical vector indicating which hypotheses are rejected  
`errorControl` A Mutoss S4 class of type `errorControl`, containing the type of error controlled by the function and the level alpha.

**Author(s)**

WerftWiebke

**References**

Benjamini, Y. and Yekutieli, D. (2001). The control of the false discovery rate in multiple testing under dependency. *Annals of Statistics*, 29(4):1165-1188.

**Examples**

```
alpha <- 0.05
p <-c(runif(10, min=0, max=0.01), runif(10, min=0.9, max=1))
result <- BY(p, alpha)
result <- BY(p, alpha, silent=TRUE)
```

---

calculateBetaAdjustment

*Calculating the beta adjustment factor for the asymptotically optimal rejection curve.*

---

**Description**

Calculates the beta to adjust the asymptotically optimal rejection curve used by the function aorc() for a finite sample size. Then aorc(..., betaAdjustment = beta) controls the FDR also in the finite sample situation.

**Usage**

```
calculateBetaAdjustment(n, startIDX_SUD, alpha, silent=FALSE,
  initialBeta=1, maxBinarySteps=50, tolerance=1e-04)
```

**Arguments**

|                |  |
|----------------|--|
| n              | Number of tests for which the adjusted beta should be calculated.                    |
| startIDX_SUD   | Starting index of the step-up-down procedure   |
| alpha          | The level at which the FDR shall be controlled.                                      |
| silent         | If true any output on the console will be suppressed.                                |
| initialBeta    | Initial beta.  |
| maxBinarySteps | Maximum number of steps that will be performed.                                      |
| tolerance      | The tolerance to search for an upper FDR bound element in [alpha - tolerance, alpha] |



**Details**

The asymptotically optimal rejection curve, denoted by  $f(t)$ , does not provide finite control of the FDR. `calculateBetaAdjustment()` calculates a factor, denoted by  $\beta$ , such that  $(1 + \beta/n) * f(t)$  provides finite control of the FDR.

The  $\beta$  is calculated with the bisection approach. Assume there are  $\beta_1$  and  $\beta_2$  such that the choosing  $\beta_1$  controls the FDR and  $\beta_2$  not, then the optimal  $\beta$  lies in  $[\beta_2, \beta_1]$ . If the choice  $(\beta_2 + \beta_1)/2$  controls the FDR, the optimal FDR lies in  $[(\beta_2 + \beta_1)/2, \beta_1]$  and so on.

**Value**

The adjustment factor that is needed to ensure control of the FDR with the adjusted asymptotically optimal rejection curve at the specified level and sample size.

**Author(s)**

MarselScheer

---

compareMutoss

*Functions for comparing outputs of different procedures.*

---

**Description**

Functions for comparing outputs of different procedures.

**Usage**

```
compareMutoss(...)
mu.compare.adjusted(comparison.list, identify.check=F)
mu.compare.critical(comparison.list, identify.check=F)
mu.compare.summary(comparison.list)
```

**Arguments**

`...` An arbitrary number of Mutoss class objects.

`comparison.list` The output of the `compareMutoss` function.

`identify.check` Logical parameter specifying if hypotheses should be identified on the output plots.

**Details**

These functions are used to compare the results of different multiple comparisons procedures stored as Mutoss class objects. `compareMutoss` takes as input an arbitrary number of Mutoss objects and arranges them in a simple list objects (non S4). `mu.compare.adjusted`, `mu.compare.critical` and `mu.compare.summary` take the output of the `compareMutoss` and plots or summerize the results textually or graphically.

**Value**

compareMutoss Returns a list with the following components:  
**types**: Character vector of error types corresponding to each procedure.  
**rates**: Numeric vector of error rates used for each procedure.  
**pi.nulls**: Numeric vector of estimates of the proportion of true null hypothesis if available.  
**raw.pValues**: The raw p-values used for each procedure.  
**adjusted.pvals**: Data frame with columns holding procedure specific adjusted p values.  
**criticalValue**: Data frame with columns holding the critical values corresponding to each procedure and error rate.  
**rejections**: Data frame with columns holding logical vectors of rejected hypotheses (TRUE for rejected).

mu.compare.adjusted  
 Creates a plot with the adjusted p-values for each procedure.

mu.compare.critical  
 Creates a plot with the critical values for each procedure and error rate.

mu.compare.summary  
 Creates a short textual summary for comparing results of different procedures.

**Author(s)**

Jonathan Rosenblatt

**Examples**

```
# TODO: EXAMPLE PROBLEMS
## Not run:
\dontrun{Creating several Mutoss class objects}
mu.test.obj.1 <- mutoss.apply(new(Class="Mutoss",
                                pValues=runif(10)),
                              f=bonferroni,
                              label="Bonferroni Correction",
                              alpha=0.05,
                              silent=T)
mu.test.obj.2 <- mutoss.apply(new(Class="Mutoss",
                                pValues=runif(10)),
                              f=holm,
                              label="Holm's step-down-procedure",
                              alpha=0.05,
                              silent=T)
mu.test.obj.3 <- mutoss.apply(new(Class="Mutoss",
                                pValues=runif(10)),
                              f=aorc,
                              label="Asymptotically optimal rejection curve",
                              alpha=0.05,
                              startIDX_SUD = 1,
                              silent=T)
\dontrun{Trying to coercing a non-Mutoss object}
```

```

compareMutoss(1)
\dontrun{ Coercing several objects into a list}
compare.1<- compareMutoss(mu.test.obj.1, mu.test.obj.2)
compare.2<- compareMutoss(mu.test.obj.1, mu.test.obj.2, mu.test.obj.3)
\dontrun{Plotting the adjusted pvalues. Identification available.}
mu.compare.adjusted(compare.1, T)
mu.compare.adjusted(compare.2, T)
\dontrun{Plotting the critical values. Identification available.}
mu.compare.critical(compare.1, T)
mu.compare.critical(compare.2, T)
\dontrun{Showing a textual sumary}
mu.compare.summary(compare.1)
mu.compare.summary(compare.2)

## End(Not run)

```

---

|                    |                           |
|--------------------|---------------------------|
| ErrorControl-class | <i>Class ErrorControl</i> |
|--------------------|---------------------------|

---

### Description

This class holds the information related to the error rate

### Slots

**type:** a character specifying the error rate. For example FWER, FDR, ...

**alpha:** the level at which the error rate shall be controlled

**k:** an additional parameter for generalised FWER

**q:** an additional parameter for FDX (false discovery exceedance)

### Author(s)

MuToss-Coding Team

---

|                   |   |
|-------------------|---|
| fisher22.marginal | <i>Fisher (2x2) table association analysis for calculating all marginal p-values...</i> |
|-------------------|---|

---

### Description

Fisher (2x2) table association analysis for calculating all marginal p-values

### Usage

```
fisher22.marginal(data, model)
```

**Arguments**

data            A tensor of dimension (2x2xm) where m is the number of endpoints (genes, etc.)  
 model           A model object indicating that this type of analysis shall be performed

**Value**

A list containing the m marginal p-values

**Author(s)**

ThorstenDickhaus

---

|               |  |
|---------------|--|
| fisher22_fast | <i>Fisher (2x2) table association analysis for calculating one (marginal) p-value...</i> |
|---------------|--|

---

**Description**

Fisher (2x2) table association analysis for calculating one (marginal) p-value

**Usage**

```
fisher22_fast(obs, epsilon)
```

**Arguments**

obs            The observed (2x2) table  
 epsilon        A threshold for comparing real numbers to zero

**Value**

A list containing:

nonrand\_p      The non-randomized (conservative) p-value  
 rand\_p         The randomized (non-conservative) p-value  
 prob\_table     The conditional probability of the observed table (given the marginals)

**Author(s)**

ThorstenDickhaus

---

|                   |  |
|-------------------|--|
| fisher23.marginal | <i>Fisher-type (2x3) table association analysis for calculating all marginal p-values...</i> |
|-------------------|--|

---

**Description**

Fisher-type (2x3) table association analysis for calculating all marginal p-values

**Usage**

```
fisher23.marginal(data, model)
```

**Arguments**

|       |  |
|-------|--|
| data  | A tensor of dimension (2x3xm) where m is the number of endpoints (genes, etc.) |
| model | A model object indicating that this type of analysis shall be performed        |

**Value**

A list containing the m marginal p-values

**Author(s)**

ThorstenDickhaus

---

|               |   |
|---------------|---|
| fisher23_fast | <i>Fisher-type (2x3) table association analysis for calculating one (marginal) p-value...</i> |
|---------------|---|

---

**Description**

Fisher-type (2x3) table association analysis for calculating one (marginal) p-value

**Usage**

```
fisher23_fast(obs, epsilon)
```

**Arguments**

|         |  |
|---------|--|
| obs     | The observed (2x3) table                       |
| epsilon | A threshold for comparing real numbers to zero |

**Value**

A list containing:

|            |   |
|------------|---|
| nonrand_p  | The non-randomized (conservative) p-value                               |
| rand_p     | The randomized (non-conservative) p-value                               |
| prob_table | The conditional probability of the observed table (given the marginals) |

**Author(s)**

Thorsten Dickhaus

---

|                |                        |
|----------------|------------------------|
| ftest.marginal | <i>Marginal F test</i> |
|----------------|------------------------|

---

**Description**

The robust version uses the Kruskal-Wallis test, otherwise a F-test will be performed.

**Usage**

```
ftest.marginal(data, model, robust)
```

**Arguments**

|        |   |
|--------|---|
| data   | the data set  |
| model  | the result of a call to <code>ftest.model(classlabel)</code>                            |
| robust | a logical variable indicating whether a F-test or a Kruskal-Wallis test should be used. |

**Details**

A classlabel needs to be provided to distinguish k sample groups.

**Value**

pValues            A numeric vector containing the unadjusted pValues

**Author(s)**

MuToss-Coding Team

**References**

Kruskal, W.H. und Wallis, W.A. (1952). Use of ranks in one-criterion variance analysis. *JASA*, 47:583-621

gao

*Xin Gao's non-parametric multiple test procedure is applied to Data.***Description**

Xin Gao's non-parametric multiple test procedure is applied to Data. The procedure controls the FWER in the strong sense. Here, only the Many-To-One comparisons are computed.

**Usage**

```
gao(formula, data, alpha=0.05, control=NULL, silent=FALSE)
gao.wrapper(model, data, alpha, control)
```

**Arguments**

|         |   |
|---------|---|
| formula | Formula defining the statistical model, containing the response and the factors                               |
| model   | Model with formula, containing the response and the factors   |
| data    | Dataset containing the response and the grouping factor   |
| alpha   | The level at which the FWER shall be controlled. By default it is alpha=0.05.                                 |
| silent  | If true any output on the console will be suppressed.   |
| control | The control group for the Many-To-One comparisons. By default it is the first group in lexicographical order. |

**Details**

This function computes Xin Gao's nonparametric multiple test procedures in an unbalanced one way layout. It is based upon the following purely nonparametric effects: Let  $F_i$  denote the distribution function of sample  $i, i = 1, \dots, a$ , and let  $G$  denote the mean distribution function of all distribution functions ( $G = 1/a \sum_i F_i$ ). The effects  $p_i = \int G dF_i$  are called unweighted relative effects. If  $p_i > 1/2$ , the random variables from sample  $i$  tend (stochastically) to larger values than any randomly chosen number from the whole experiment. If  $p_i = 1/2$ , there is no tendency to smaller nor larger values. However, this approach tests the hypothesis  $H_0^F : F_1 = F_j, j = 2, \dots, a$  formulated in terms of the distribution functions, simultaneously.

**Value**

A list containing:

|               |  |
|---------------|--|
| adjPValues    | A numeric vector containing the adjusted pValues   |
| rejected      | A logical vector indicating which hypotheses are rejected  |
| confIntervals | A matrix containing the estimates and the lower and upper confidence bound   |
| errorControl  | A Mutoss S4 class of type errorControl, containing the type of error controlled by the function and the level alpha. |

**Author(s)**

Frank Konietschke

**References**

Gao, X. et al. (2008). Nonparametric multiple comparison procedures for unbalanced one-way factorial designs. *Journal of Statistical Planning and Inference* 77, 2574-2591. *n* The FWER is controlled by using the Hochberg adjustment (Hochberg, Y. (1988). A sharper Bonferroni procedure for multiple tests of significance. *Biometrika* 75, 800-802.)

**Examples**

```
x=c(rnorm(40))
f1=c(rep(1,10),rep(2,10),rep(3,10),rep(4,10))
my.data <- data.frame(x,f1)
result <- gao(x~f1,data=my.data, alpha=0.05,control=2, silent=FALSE)
result <- gao(x~f1,data=my.data, alpha=0.05,control=2, silent=TRUE)
result <- gao(x~f1,data=my.data, alpha=0.05)
```

---

gatherParameters

*Extracts the parameters from the simulation()-Object*


---

**Description**

Basically, it is a helper function for gatherStatistics(). It extracts the parameters from the simObject\$results and creates a data.frame from this. Every used parameter gets its own column. Every row corresponds to one object in simObject\$results. If a object A from simObject\$results does not contain a parameter P that another object B does, then the row for object A will have "" in the column for the parameter P.

**Usage**

```
gatherParameters(simObject)
```

**Arguments**

simObject      An object returned by simulation()

**Value**

A data.frame with rows for every object in simObject\$results and columns for the used parameter.

**Author(s)**

MarselScheer



**Examples**

```

#' # this function generates pValues
myGen <- function(n, n0) {
  list(procInput=list(pValues = c(runif(n-n0, 0, 0.01),
    runif(n0))), groundTruth = c(rep(FALSE, times=n-n0), rep(TRUE, times=n0)))
}

# need some simulation()-Object I can work with
sim <- simulation(replications = 3, list(funName="myGen", fun=myGen, n=200, n0=c(50,100)),
  list(list(funName="BH", fun=function(pValues, alpha) BH(pValues, alpha, silent=TRUE),
    alpha=c(0.25, 0.5)),
  list(funName="holm", fun=holm, alpha=c(0.25, 0.5), silent=TRUE)))

gatherParameters(sim)

```

---

|                  |  |
|------------------|--|
| gatherStatistics | <i>Gathering statistics from simulation()-Object</i> |
|------------------|--|

---

**Description**

This function facilitates gathering statistics from the object returned by simulation().

**Usage**

```
gatherStatistics(simObject, listOfStatisticFunctions,
  listOfAvgFunctions)
```

**Arguments**

**simObject**      An object returned by simulation()  
**listOfStatisticFunctions**  
                   List of statistics that shall be calculated for the elements in simObject  
**listOfAvgFunctions**  
                   List of functions that will be used to summarize/average the calculated statistics for all elements in simObject\$results with the same parameter constellation. If this is argument is missing no averaging will be done. Instead the resulting data.frame will keep one row for every object in simObject\$results.

**Details**

For every simulation()-Object in simObject\$results all statistics in listOfStatisticFunctions are calculated. If in addition listOfAvgFunctions is provided then the statistics of the objects that have the same parameter constellation are averaged by the functions in listOfAvgFunctions. The resulting data.frame will then keep only one row for every parameter constellation.

**Value**

`statisticDF` A data.frame that can be of two different kinds.

If `listOfAvgFunctions` is provided, then the resulting data.frame will have a row for every parameter constellation that occurs in the `simObject$results`. There will be columns for every parameter used. Also `length(listOfAvgFunctions) * length(listOfStatisticFunctions)` additional columns will be created. Every statistic is calculated for every applied procedure and then all values that belong to a specific parameter constellation are "averaged" by applying all function from `listOfAvgFunctions`. For example, suppose `FDP` is a function from `listOfStatisticFunctions` that calculates the realized false discovery proportion, that is number of true hypotheses that were rejected divided by the number of all rejected hypotheses. Also suppose `mean` and `sd` are functions in `listOfAvgFunctions`, then the mean and the standard deviation of the statistic `FDP` are calculated.

If `listOfAvgFunctions` is not provided, then the resulting data.frame will have a row for every `simObject$results`. Every parameter will have its own column. Additional columns for every function in `listOfStatisticFunction` will be created.

**Author(s)**

MarselScheer

**Examples**

```
#' # this function generates pValues
myGen <- function(n, n0) {
  list(procInput=list(pValues = c(runif(n-n0, 0, 0.01), runif(n0))),
        groundTruth = c(rep(FALSE, times=n-n0), rep(TRUE, times=n0)))
}

# need some simulation()-Object I can work with
sim <- simulation(replications = 10, list(funName="myGen", fun=myGen, n=200, n0=c(50,100)),
  list(list(funName="BH", fun=function(pValues, alpha) BH(pValues, alpha, silent=TRUE),
    alpha=c(0.25, 0.5)),
  list(funName="holm", fun=holm, alpha=c(0.25, 0.5),silent=TRUE)))

# Make my own statistic function
NumberOfType1Error <- function(data, result) sum(data$groundTruth * result$rejected)

# Get now for every object in sim$results one row with the number of Type 1 Errors
result.all <- gatherStatistics(sim, list(NumOfType1Err = NumberOfType1Error))

# Average over all sim$results-Objects with common parameters
result1 <- gatherStatistics(sim, list(NumOfType1Err = NumberOfType1Error), list(MEAN = mean))
print(result1)
result2 <- gatherStatistics(sim, list(NumOfType1Err = NumberOfType1Error),
  list(q05 = function(x) quantile(x, probs=0.05),
    MEAN = mean, q95 = function(x) quantile(x, probs=0.95)))
print(result2)
```

```
# create some plots
require(lattice)
histogram(~NumOfType1Err | method*alpha, data = result.all$statisticDF)
barchart(NumOfType1Err.MEAN ~ method | alpha, data = result2$statisticDF)
```

hochberg

*Hochberg (1988) step-up procedure***Description**

The Hochberg step-up procedure is based on marginal p-values. It controls the FWER in the strong sense under joint null distributions of the test statistics that satisfy Simes' inequality.

**Usage**

```
hochberg(pValues, alpha, silent=FALSE)
```

**Arguments**

|         |   |
|---------|---|
| pValues | The used raw pValues.                                 |
| alpha   | The level at which the FDR shall be controlled.       |
| silent  | If true any output on the console will be suppressed. |

**Details**

The Hochberg procedure is more powerful than Holm's (1979) procedure, but the test statistics need to be independent or have a distribution with multivariate total positivity of order two or a scale mixture thereof for its validity (Sarkar, 1998). Both procedures use the same set of critical values  $c(i)=\alpha/(m-i+1)$ . Whereas Holm's procedure is a step-down version of the Bonferroni test, and Hochberg's is a step-up version of the Bonferroni test. Note that Holm's method is based on the Bonferroni inequality and is valid regardless of the joint distribution of the test statistics.

**Value**

A list containing:

|                |  |
|----------------|--|
| adjPValues     | A numeric vector containing the adjusted pValues   |
| rejected       | A logical vector indicating which hypotheses are rejected  |
| criticalValues | A numeric vector containing critical values used in the step-up-down test  |
| errorControl   | A Mutoss S4 class of type errorControl, containing the type of error controlled by the function and the level alpha. |

**Author(s)**

WerftWiebke

## References

Hochberg, Y. (1988). A sharper Bonferroni procedure for multiple tests of significance. *Biometrika*, 75:800-802.*n*

Huang, Y. and Hsu, J. (2007). Hochberg's step-up method: cutting corners off Holm's step-down method. *Biometrika*, 94(4):965-975.

## Examples

```
alpha <- 0.05
p <-c(runif(10, min=0, max=0.01), runif(10, min=0.9,max=1))
result <- hochberg(p, alpha)
result <- hochberg(p, alpha, silent=TRUE)
```

---

holm

*Holm's (1979) step-down-procedure*

---

## Description

Holm's step-down-procedure is applied to pValues. It controls the FWER in the strong sense under arbitrary dependency.

## Usage

```
holm(pValues, alpha, silent=FALSE)
```

## Arguments

|         |   |
|---------|---|
| pValues | pValues to be used. They can have arbitrary dependency structure. |
| alpha   | The level at which the FWER shall be controlled                   |
| silent  | If true any output on the console will be suppressed.             |

## Details

Holm's procedure uses the same critical values as Hochberg's procedure, namely  $c(i)=\alpha/(m-i+1)$ , but is a step-down version while Hochberg's method is a step-up version of the Bonferroni test. Holm's method is based on the Bonferroni inequality and is valid regardless of the joint distribution of the test statistics, whereas Hochberg's method relies on the assumption that Simes' inequality holds for the joint null distribution of the test statistics. If this assumption is met, Hochberg's step-up procedure is more powerful than Holm's step-down procedure.

## Value

A list containing:

|            |   |
|------------|---|
| adjPValues | A numeric vector containing the adjusted pValues          |
| rejected   | A logical vector indicating which hypotheses are rejected |

`criticalValues` A numeric vector containing critical values used in the step-down test

`errorControl` A Mutoss S4 class of type `errorControl`, containing the type of error controlled by the function and the level `alpha`.

**Author(s)**

MarselScheer

**References**

S. Holm (1979). A simple sequentially rejective multiple test procedure. *Scand. J. Statist.* Vol. 6, 65-70. *n*

Huang, Y. and Hsu, J. (2007). Hochberg's step-up method: cutting corners off Holm's step-down method. *Biometrika*, 94(4):965-975.

**Examples**

```
r <- c(runif(50), runif(50, 0, 0.01))
result <- homl(r, 0.05)
result <- homl(r, 0.05, silent = TRUE)
```

---

hommel

*Hommel's (1988) step-up-procedure*

---

**Description**

Hommel's step-up-procedure.

**Usage**

```
hommel(pValues, alpha, silent=FALSE)
```

**Arguments**

`pValues` pValues to be used. They need a independent structure.

`alpha` The level at which the FWER should be controlled

`silent` Logical. If true, any output on the console will be suppressed.

**Details**

The method is applied to p-values. It controls the FWER in the strong sense when the hypothesis tests are independent or when they are non-negatively associated.

The method is based upon the closure principle and the Simes test.

**Value**

A list containing:

adjPValues      A numeric vector containing the adjusted pValues  
 rejected        A logical vector indicating which hypotheses are rejected  
 criticalValues   A numeric vector containing critical values used in the step-up-down test.  
 errorControl    A Mutoss S4 class of type errorControl, containing the type of error controlled by the function and the level alpha.

**Author(s)**

HackNiklas

**References**

G. Hommel (1988). A stagewise rejective multiple test procedure based on a modified Bonferroni test. *Biometrika* 75, pp. 383-386

**Examples**

```
pval <- c(runif(50), runif(50, 0, 0.01))
result <- hommel(pval, 0.05)
result <- hommel(pval, 0.05, silent = TRUE)
```

---

indepBR

*Blanchard-Roquain (2009) 1-stage adaptive step-up*

---

**Description**

Blanchard-Roquain (2009) 1-stage adaptive step-up

**Usage**

```
indepBR(pValues, alpha, lambda=1, silent=FALSE)
```

**Arguments**

pValues        the used p-values (assumed to be independent)  
 alpha         the level at which the FDR should be controlled.  
 lambda        parameter of the procedure, should belong to (0, 1/alpha) (lambda=1 default)  
 silent        if true any output on the console will be suppressed.

**Details**

This is a step-up procedure with critical values

$$C_i = \alpha * \min( i * ( 1 - \lambda * \alpha ) / ( m - i + 1 ) , \lambda )$$

where  $\alpha$  is the level at which FDR should be controlled and  $\lambda$  an arbitrary parameter belonging to  $(0, 1/\alpha)$  with default value 1. This procedure controls FDR at the desired level when the p-values are independent.

**Value**

A list containing:

|                |  |
|----------------|--|
| rejected       | A logical vector indicating which hypotheses are rejected  |
| criticalValues | A numeric vector containing critical values used in the step-up-down test  |
| errorControl   | A Mutoss S4 class of type errorControl, containing the type of error controlled by the function and the level $\alpha$ . |

**Author(s)**

GillesBlanchard

**References**

Blanchard, G. and Roquain, E. (2009) Adaptive False Discovery Rate Control under Independence and Dependence *Journal of Machine Learning Research* 10:2837-2871.

---

jointCDF.orderedUnif *Joint cumulative distribution function of order statistics of n iid. U(0,1)-distributed random variables*

---

**Description**

Calculates the joint cumulative distribution function of order statistics of n iid. U(0,1)-distributed random variables at argument vec. Because of numerical issues n should not be greater than 100.

**Usage**

```
jointCDF.orderedUnif(vec)
```

**Arguments**

|     |   |
|-----|---|
| vec | a numeric vector. The length of the vector also determines the number of random variables considered. |
|-----|---|

**Details**

Following Shorack, Wellner (1986) or Finner, Roters (2002) by applying Bolshev's recursion the joint distribution is calculated.

**Value**

The return value is the following probability  $P(U_{(1:n)} \leq \text{vec}[1], \dots, U_{(n:n)} \leq \text{vec}[n])$ , where  $U_1, \dots, U_n$  are assumed to be iid. uniformly distributed on  $[0,1]$ . The  $i$ -th ordered value is denoted by  $U_{(i:n)}$  and  $n$  equals  $\text{length}(\text{vec})$

**Author(s)**

MarselScheer

**References**

Shorack, G. R. and Wellner, J. A. (1986). Empirical Processes with Applications to Statistics. Wiley, New York.

Finner, H. and Roters, M. (2002). Multiple hypotheses testing and expected type I errors. Ann. Statist. 30, 220-238.

---

linearStepUp

*Linear Step Up Service Function...*

---

**Description**

Linear Step Up Service Function

**Usage**

`linearStepUp(sorted, q, m, adjust=FALSE, m0=m, pi0, constant=1)`

**Arguments**

|                       |  |
|-----------------------|--|
| <code>sorted</code>   | Numeric vector of sorted pvalues   |
| <code>q</code>        | Error rate to control for.   |
| <code>m</code>        | Number of hypothesis tested.   |
| <code>adjust</code>   | Logical value for p-value adjustmet (unusable).  |
| <code>m0</code>       | Known or estimated number of true null hypotheses.   |
| <code>pi0</code>      | Known or estimated proportion of true null hypothesis. Is redundant when <code>m0</code> is specified. |
| <code>constant</code> | A Scaling constant for the denominator of the critical values.   |

**Details**

A Mutoss service function called by other procedures.



**Value**

A list containing the following objects:

|        |   |
|--------|---|
| Cutoff | The largest p-value of rejected hypotheses.   |
| Pvals  | A data frame containing the original p-values, critical values, adjusted p-values and rejections. |

**Author(s)**

JonathanRosenblatt

---

|               |  |
|---------------|--|
| mu.test.class | <i>Test for compatible classes for comparison...</i> |
|---------------|--|

---

**Description**

Test for compatible classes for comparison

**Usage**

```
mu.test.class(classes)
```

**Arguments**

classes      A list of class names.

**Details**

Internal function of muToss package. Takes list of classes and tests if they are all muToss.

**Value**

True if all classes are muToss. False otherwise.

**Author(s)**

MuToss-Coding Team.

---

|              |   |
|--------------|---|
| mu.test.name | <i>Tests if all hypotheses have the same names.</i> |
|--------------|---|

---

**Description**

Tests if all hypotheses have the same names.

**Usage**

```
mu.test.name(hyp.names)
```

**Arguments**

hyp.names      Character vector of hypotheses names.

**Details**

Internal muToss function.

**Value**

Gives a notice when hypotheses names in different procedures are found different.

**Author(s)**

MuToss-Coding Team

---

|               |   |
|---------------|---|
| mu.test.rates | <i>Tests that different procedures used the same error rates.</i> |
|---------------|---|

---

**Description**

Tests that different procedures used the same error rates.

**Usage**

```
mu.test.rates(rates)
```

**Arguments**

rates            Numeric vector error rates extracted fomr muToss objects.

**Details**

Internal muToss function.

**Value**

A notice if error rates differ.

**Author(s)**

MuToss-Coding Team.

---

`mu.test.same.data`      *Tests if the same pvalues were used by different procedures.*

---

**Description**

Tests if the same pvalues were used by different procedures.

**Usage**

```
mu.test.same.data(pvals)
```

**Arguments**

`pvals`              Data frame of p-values used by each procedure.

**Details**

Internal muToss function.

**Value**

Stops if different data was used by different procedures.

**Author(s)**

MuToss-Coding Team

---

|              |  |
|--------------|--|
| mu.test.type | <i>Tests that different procedures use the same error types.</i> |
|--------------|--|

---

**Description**

Tests that different procedures use the same error types.

**Usage**

```
mu.test.type(types)
```

**Arguments**

types            Character vector of error types extracted from muToss objects.

**Details**

Internal muToss function.

**Value**

Returns a notice if error types differ.

**Author(s)**

MuToss-Coding Team.

---

|                  |  |
|------------------|--|
| multcomp.wrapper | <i>Simultaneous confidence intervals for arbitrary parametric contrasts in unbalanced one-way layouts.</i> |
|------------------|--|

---

**Description**

Simultaneous confidence intervals for arbitrary parametric contrasts in unbalanced one-way layouts. The procedure controls the FWER in the strong sense.

**Usage**

```
multcomp.wrapper(model, hypotheses, alternative, rhs=0, alpha, factorC)
```

**Arguments**

|             |  |
|-------------|--|
| model       | a fitted model, for example an object returned by lm, glm, or aov etc. It is assumed that coef and vcov methods are available for model. |
| hypotheses  | a specification of the linear hypotheses to be tested.   |
| alternative | a character string specifying the alternative hypothesis, must be one of 'two.sided' (default), 'greater' or 'less'.                     |
| rhs         | an optional numeric vector specifying the right hand side of the hypothesis.   |
| alpha       | the significance level   |
| factorC     | character string, specifying the factor variable of interest   |

**Details**

this function, it is possible to compute simultaneous confidence for arbitrary parametric contrasts in the unbalanced one way layout. Moreover, it computes p-values. The simultaneous confidence intervals are computed using multivariate t-distribution.

**Value**

A list containing:

|               |  |
|---------------|--|
| adjPValues    | A numeric vector containing the adjusted pValues   |
| rejected      | A logical vector indicating which hypotheses are rejected  |
| confIntervals | A matrix containing the estimates and the lower and upper confidence bound                       |
| errorControl  | A Mutoss S4 class of type errorControl, containing the type of error controlled by the function. |

**Author(s)**

MuToss-Coding Team

**Examples**

```
data(warpbreaks)
# Tukey contrast on the levels of the factor 'Tension'

multcomp.wrapper(aov(breaks ~ tension, data = warpbreaks),
  hypotheses = "Tukey", alternative="two.sided", factorC="tension",alpha=0.05)

# Williams contrast on 'Tension'
multcomp.wrapper(aov(breaks ~ tension, data = warpbreaks),
  hypotheses= "Williams", alternative="two.sided",alpha=0.05,factorC="tension")

# Userdefined contrast matrix
K <-matrix(c(-1,0,1,-1,1,0, -1,0.5,0.5),ncol=3,nrow=3,byrow=TRUE)
multcomp.wrapper(aov(breaks ~ tension, data = warpbreaks),
  hypotheses=K, alternative="two.sided",alpha=0.05,factorC="tension")

# Two-way anova
```

```

multcomp.wrapper(aov(breaks ~ tension*wool, data = warpbreaks),
  hypotheses="Tukey", alternative="two.sided",alpha=0.05,factorC="wool")
multcomp.wrapper(aov(breaks ~ tension*wool, data = warpbreaks),
  hypotheses="Tukey", alternative="two.sided",alpha=0.05,factorC="tension")
multcomp.wrapper(aov(breaks ~ tension*wool, data = warpbreaks),
  hypotheses=K, alternative="two.sided",alpha=0.05, factorC="tension")
data(iris)
multcomp.wrapper(model=lm(Sepal.Length ~ Species, data=iris),
  hypotheses="Tukey", "two.sided",alpha=0.05, factorC="Species")
K <-matrix(c(-1,0,1,-1,1,0, -1,0.5,0.5),ncol=3,nrow=3,byrow=TRUE)
multcomp.wrapper(model=lm(Sepal.Length ~ Species, data=iris),
  hypotheses=K,"two.sided",alpha=0.05, factorC="Species")

```

---

multiple.down

*Benjamini-Krieger-Yekutieli (2006) Multi-Stage Step-Down*


---

## Description

A p-value procedure which controls the FDR for independent test statistics.

## Usage

```
multiple.down(pValues, alpha)
```

## Arguments

|         |                               |
|---------|-------------------------------|
| pValues | A numeric vector of p-values  |
| alpha   | The FDR error rate to control |

## Details

A non-linear step-down p-value procedure which control the FDR for independent test statistics and enjoys more power then other non-adaptive procedure such as the linear step-up (BH). For the case of non-independent test statistics, non-adaptive procedures such as the linear step-up (BH) or the all-purpose conservative Benjamini-Yekutieli (2001) are recommended.

## Value

A list containing:

|                |  |
|----------------|--|
| rejected       | A logical vector indicating which hypotheses are rejected  |
| criticalValues | A numeric vector containing critical values used in the step-up-down test.   |
| adjPValues     | A numeric vector containing adjusted p-values.   |
| pi0            | An estimate of the proportion of true null hypotheses among all hypotheses ( $\pi_0 = m_0/m$ ).                      |
| errorControl   | A Mutoss S4 class of type errorControl, containing the type of error controlled by the function and the level alpha. |

**Author(s)**

Jonathan Rosenblatt

**Examples**

```
pvals<- runif(100)^2
alpha<- 0.2
result<- multiple.down(pvals, alpha)
result
plot(result[['criticalValues']]~pvals)
plot(result[['adjPValues']]~pvals)
abline(v=alpha)
```

---

multiple.down.adjust    *A service function used by multiple...*

---

**Description**

A service function used by multiple.down

**Usage**

```
multiple.down.adjust(sorted, m)
```

**Arguments**

sorted            A sorted pvalue vector.  
m                 The number of hypotheses tested.

**Author(s)**

JonathanRosenblatt

---

Mutoss-class            *Class Mutoss*

---

**Description**

A Mutoss object can store the input for a multiple test procedure and also the output.

**Slots**

**data:** Raw data used in model  
**model:** link function,error family and design  
**description:** a general description  
**statistic:** for Z, T or F statistics  
**hypotheses:** of class ANY  
**hypNames:** identifiers for the hypotheses tested  
**criticalValues:** procedure-specific critical values  
**pValues:** raw p-values  
**adjPValues:** procedure-specific adjusted p-values  
**errorControl:** A Mutoss S4 class of type errorControl  
**rejected:** Logical vector of the output of a procedure at a given error rate  
**qValues:** Storey's estimates of the supremum of the pFDR  
**locFDR:** Efron's local fdr estimates  
**pi0:** Estimate of the proportion of null hypotheses  
**confIntervals:** Confidence intervals for selected parameters  
**commandHistory:** commandHistory

**Author(s)**

MuToss-Coding Team

---

mutoss.apply

*Applies a function to a Mutoss object.*

---

**Description**

Applies a function to a Mutoss object.

**Usage**

```
mutoss.apply(mutossObj, f, label = deparse(substitute(f)) , recordHistory = TRUE , ...)
```

**Arguments**

|               |  |
|---------------|--|
| mutossObj     | the Mutoss object the function should be applied to  |
| f             | the function that should be applied  |
| label         | the label affixed to all slots of the Mutoss object that are changed by the procedure, defaults to the name of parameter f |
| recordHistory | if true, the calling command is concatenated verbatim to the commandHistory slot   |
| ...           | additional parameters that are passed to the function  |



**Details**

This functions is intended for applying functions for multiplicity control on Mutoss class objects using the console and not the Mutoss GUI.

**Value**

A Mutoss object after applying the given function

mutossObj      Object of S4 class Mutoss

**Author(s)**

Kornelius Rohmeyer

**Examples**

```
newObjectBonf <- mutoss.apply(new(Class="Mutoss", pValues=runif(10)),
  f=bonferroni, label="Bonferroni Correction", alpha=0.05)
## Not run: TODO: EXAMPLE PROBLEM
newObjectHolm <- mutoss.apply(new(Class="Mutoss", pValues=runif(10)),
  f=holm, label="Holm's step-down-procedure", alpha=0.05, silent=T)

newObjectAORC <- mutoss.apply(new(Class="Mutoss", pValues=runif(10)),
  f=aorc, label="Asymptotically optimal rejection curve", alpha=0.05, startIDX_SUD = 1, silent=T)

## End(Not run)
```

---

mutoss.models

*Mutoss Models*

---

**Description**

Fisher-type (2x3) table as model for marginal hypotheses testing problems, Fisher (2x2) table as model for marginal hypotheses testing problems and others...

**Value**

A list containing the model description

---

|                            |                      |
|----------------------------|----------------------|
| <code>mutoss.plotCI</code> | <i>mutoss.plotCI</i> |
|----------------------------|----------------------|

---

**Description**

Plots the confidence intervals

**Usage**

`mutoss.plotCI(mat)`

**Arguments**

|                  |   |
|------------------|---|
| <code>mat</code> | Matrix containing the confidence interval limits. TODO specify the matrix layout. |
|------------------|---|

---

|                    |                           |
|--------------------|---------------------------|
| MutossMethod-class | <i>Class MutossMethod</i> |
|--------------------|---------------------------|

---

**Description**

A MutossMethod object describes a method that is applicable to Mutoss objects.

**Slots**

**label:** A character string that contains the label that will be shown in menus.

**errorControl:** One of the following character strings: FWER, FWER.weak, FDR, FDX, gFWER, perComparison.

**callFunction:** A character string that contains the name of the Mutoss-compatible function.

**output:** A character vector of the *possible* output of the function.

**info:** A character string with info text. Should contain small description, author, reference etc..

**assumptions:** A character vector of assumptions for this method.

**parameters:** A list of optional description of parameters - see MuToss developer handbook.

**list:** For extensions a list where you can put all your miscellaneous stuff.

---

|           |                           |
|-----------|---------------------------|
| notterman | <i>Notterman data set</i> |
|-----------|---------------------------|

---

### Description

The notterman data set is a data.frame containing 18 paired samples of 7457 gene expression values from Notterman et al. (2001).

The vector notterman.grpLabel contains 36 labels of type Tumor or Normal specifying the type of tissue for each column of the notterman data set.

The vector T.Test.tumor.vs.normal contains the resulting 7457 numeric p-values from the 7457 t-tests applied to each row of the data set.

### Usage

```
notterman
notterman.grpLabel
T.Test.tumor.vs.normal
```

### Format

- notterman - A data.frame containing 36 columns with 7457 observations
- notterman.grpLabel - A vector containing 36 labels of type Tumor or Normal
- T.Test.tumor.vs.normal - A vector containing 7457 numeric p-values

### Source

D.A. Notterman, U. Alon, A.J. Sierk, and A.J. Levine: *Transcriptional Gene Expression Profiles of Colorectal Adenoma, Adenocarcinoma, and Normal Tissue Examined by Oligonucleotide Arrays*, Cancer Research, 2001, vol. 61, pp. 3124-3130.

---

|          |   |
|----------|---|
| nparcomp | <i>Simultaneous confidence intervals for relative contrast effects...</i> |
|----------|---|

---

### Description

Simultaneous confidence intervals for relative contrast effects The procedure controls the FWER in the strong sense.

### Usage

```
nparcomp(formula, data, type=c("UserDefined", "Tukey", "Dunnett",
"Sequen", "Williams", "Changepoint", "AVE", "McDermott", "Marcus",
"UmbrellaWilliams"), control=NULL, conflevel=0.95,
alternative=c("two.sided", "less", "greater"), rounds=3,
correlation=FALSE, asy.method=c("logit", "probit", "normal",
"mult.t"), plot.simci=FALSE, info=TRUE, contrastMatrix=NULL)
```

**Arguments**

|                |  |
|----------------|--|
| formula        | A two-sided 'formula' specifying a numeric response variable and a factor with more than two levels. If the factor contains less than 3 levels, an error message will be returned  |
| data           | data A dataframe containing the variables specified in formula   |
| type           | type Character string defining the type of contrast. It should be one of "Tukey", "Dunnett", "Sequen", "Williams", "Changepoint", "AVE", "McDermott", "Marcus"   |
| control        | control Character string defining the control group in Dunnett comparisons. By default it is the first group by lexicographical ordering   |
| conflevel      | The confidence level for the 1 - conflevel confidence intervals. By default it is 0.05   |
| alternative    | Character string defining the alternative hypothesis, one of "two.sided", "less" or "greater"  |
| rounds         | Number of rounds for the numeric values of the output. By default it is rounds=3   |
| correlation    | Correlation A logical whether the estimated correlation matrix and covariance matrix should be printed   |
| asy.method     | asy.method character string defining the asymptotic approximation method, one of "logit", for using the logit transformation function, "probit", for using the probit transformation function, "normal", for using the multivariate normal distribution or "mult.t" for using a multivariate t-distribution with a Satterthwaite Approximation |
| plot.simci     | plot.simci A logical indicating whether you want a plot of the confidence intervals  |
| info           | info A logical whether you want a brief overview with informations about the output  |
| contrastMatrix | arbitrary contrast matrix given by the user  |

**Details**

With this function, it is possible to compute nonparametric simultaneous confidence intervals for relative contrast effects in the unbalanced one way layout. Moreover, it computes adjusted p-values. The simultaneous confidence intervals can be computed using multivariate normal distribution, multivariate t-distribution with a Satterthwaite Approximation of the degree of freedom or using multivariate range preserving transformations with Logit or Probit as transformation function. There is no assumption on the underlying distribution function, only that the data have to be at least ordinal numbers

**Value**

A list containing:

|               |  |
|---------------|--|
| adjPValues    | A numeric vector containing the adjusted pValues   |
| rejected      | A logical vector indicating which hypotheses are rejected  |
| confIntervals | A matrix containing the estimates and the lower and upper confidence bound                       |
| errorControl  | A Mutoss S4 class of type errorControl, containing the type of error controlled by the function. |

**Author(s)**

FrankKonietschke

**Examples**

```
## Not run: # TODO Check this example and set a seed!
grp <- rep(1:5,10)
x <- rnorm(50, grp)
dataframe <- data.frame(x,grp)
# Williams Contrast
nparcomp(x ~grp, data=dataframe, asy.method = "probit",
type = "Williams", alternative = "two.sided", plot.simci = TRUE, info = TRUE)

# Dunnett Contrast
nparcomp(x ~grp, data=dataframe, asy.method = "probit",control=1,
type = "Dunnett", alternative = "two.sided", plot.simci = TRUE, info = TRUE)

# Dunnett dose 3 is baseline
nparcomp(x ~grp, data=dataframe, asy.method = "probit",
type = "Dunnett", control = "3",alternative = "two.sided",
plot.simci = TRUE, info = TRUE)

## End(Not run)
```

nparcomp.wrapper

*Simultaneous confidence intervals for relative contrast effects...***Description**

Simultaneous confidence intervals for relative contrast effects The procedure controls the FWER in the strong sense.

**Usage**

```
nparcomp.wrapper(model, data, hypotheses, alpha, alternative,
  asy.method)
```

**Arguments**

|             |  |
|-------------|--|
| model       | A two-sided formula specifying a numeric response variable and a factor with more than two levels.   |
| data        | A dataframe containing the variables specified the model   |
| hypotheses  | Character string defining the type of contrast. It should be one of "Tukey", "Dunnett", "Sequen", "Williams", "Changepoint", "AVE", "McDermott", "Marcus". |
| alpha       | the significance level   |
| alternative | Character string defining the alternative hypothesis, one of "two.sided", "less" or "greater"  |

`asy.method` A character string defining the asymptotic approximation method, one of "logit", for using the logit transformation function, "probit", for using the probit transformation function, "normal", for using the multivariate normal

### Details

With this function, it is possible to compute nonparametric simultaneous confidence intervals for relative contrast effects in the unbalanced one way layout. Moreover, it computes adjusted p-values. The simultaneous confidence intervals can be computed using multivariate normal distribution, multivariate t-distribution with a Satterthwaite Approximation of the degree of freedom or using multivariate range preserving transformations with Logit or Probit as transformation function. There is no assumption on the underlying distribution function, only that the data have to be at least ordinal numbers

### Value

A list containing:

`adjPValues` A numeric vector containing the adjusted pValues  
`rejected` A logical vector indicating which hypotheses are rejected  
`confIntervals` A matrix containing the estimates and the lower and upper confidence bound  
`errorControl` A Mutoss S4 class of type `errorControl`, containing the type of error controlled by the function.

### Author(s)

FrankKonietschke

---

`onesamp.marginal` *Marginal one sample test*

---

### Description

The robust version uses the Wilcoxon-Mann-Whitney test, otherwise a t-test will be performed.

### Usage

```
onesamp.marginal(data, robust, alternative, psi0)
```

### Arguments

`data` the data set  
`robust` a logical variable indicating whether a t-test or a Wilcoxon-Mann-Whitney test should be used.  
`alternative` a character string specifying the alternative hypothesis, must be one of `two.sided`, `greater` or `less`  
`psi0` a numeric that defines the hypothesized null value

**Value**

pValues            A numeric vector containing the unadjusted pValues

**Author(s)**

MuToss-Coding Team

**References**

Wilcoxon, F. (1945). Individual Comparisons by Ranking Methods. *Biometrics Bulletin* 1:80-83.  
 Mann, H. and Whitney, D. (1947). On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics* 18:50-60  
 Student (1908). The probable error of a mean. *Biometrika*, 6(1):1-25.

---

oracleBH                            *Benjamini-Hochberg (2000) oracle linear step-up Procedure...*

---

**Description**

Benjamini-Hochberg (2000) oracle linear step-up Procedure

**Usage**

oracleBH(pValues, alpha, pi0, silent=FALSE)

**Arguments**

pValues            pValues to be used  
 alpha              the level at which the FWER should be controlled  
 pi0                 miraculously known number of true null hypotheses  
 silent              Logical, if true any output on the console will be suppressed.

**Details**

Knowledge of the number of true null hypotheses ( $m_0$ ) can be very useful to improve upon the performance of the FDR controlling procedure. For the oracle linear step-up procedure we assume that  $m_0$  were given to us by an 'oracle', the linear step-up procedure with  $q_0 = q \cdot m/m_0$  would control the FDR at precisely the desired level  $q$  in the independent and continuous case, and would then be more powerful in rejecting hypotheses for which the alternative holds.

**Value**

A list containing:

adjPValues        A numeric vector containing the adjusted pValues  
 rejected          A logical vector indicating which hypotheses are rejected  
 criticalValues    A numeric vector containing critical values used in the step-up-down test.  
 errorControl     A Mutoss S4 class of type errorControl, containing the type of error controlled by the function and the level alpha.

**Author(s)**

HackNiklas

**Examples**

```
pval <- c(runif(50), runif(50, 0, 0.01))
result <- oracleBH(pValues=pval,alpha=0.05,psi0=0.85)
```

---

|                              |  |
|------------------------------|--|
| <code>paired.marginal</code> | <i>Marginal paired two sample test</i> |
|------------------------------|--|

---

**Description**

The robust version uses the Wilcoxon signed rank test, otherwise a paired t-test will be performed.

**Usage**

```
paired.marginal(data, model, robust, alternative, psi0, equalvar)
```

**Arguments**

|                          |   |
|--------------------------|---|
| <code>data</code>        | the data set  |
| <code>model</code>       | the result of a call to <code>paired.model(classlabel)</code>   |
| <code>robust</code>      | a logical variable indicating whether a paired t-test or a Wilcoxon signed rank test should be used.  |
| <code>alternative</code> | a character string specifying the alternative hypothesis, must be one of <code>two.sided</code> , <code>greater</code> or <code>less</code> |
| <code>psi0</code>        | a numeric that defines the hypothesized null value  |
| <code>equalvar</code>    | a logical variable indicating whether to treat the two variances as being equal   |

**Details**

A vector of classlabels needs to be provided to distinguish the two paired groups. The arrangement of group indices does not matter, as long as the columns are arranged in the same corresponding order between groups. For example, if group 1 is coded as 0 and group 2 is coded as 1, for 3 pairs of data, it does not matter if the classlabel is coded as (0,0,0,1,1,1) or (1,1,1,0,0,0) or (0,1,0,1,0,1) or (1,0,1,0,1,0), the paired differences between groups will be calculated as `group2 - group1`.

**Value**

`pValues`            A numeric vector containing the unadjusted pValues

**Author(s)**

MuToss-Coding Team



**References**

Wilcoxon, F. (1945). Individual Comparisons by Ranking Methods. *Biometrics Bulletin* 1:80-83.

---

|               |  |
|---------------|--|
| printRejected | <i>Internal MuTossProjekt-Function</i> |
|---------------|--|

---

**Description**

Generates standard output for pValues, rejected and adjustedPValues.

**Usage**

```
printRejected(rejected, pValues=NULL, adjPValues=NULL)
```

**Arguments**

|            |   |
|------------|---|
| rejected   | logical Vector indicating which pValue is rejected. |
| pValues    | the used pValues.                                   |
| adjPValues | the adjusted pValues.                               |

**Details**

It generates an output on the console with the number of hypotheses (number of pValues) and the number of rejected hypotheses (number of rejected pValues). Further a data.frame is constructed, one column containing the rejected pValues, one the index number of the rejected pValues and if given one column with the corresponding adjusted pValues.

**Author(s)**

MarselScheer

---

|             |  |
|-------------|--|
| pval2locfdr | <i>Strimmer et al.'s fdrtool-based local fdr</i> |
|-------------|--|

---

**Description**

The function pval2locfdr takes a vector of p-values and estimates for each case the local fdr.

**Usage**

```
pval2locfdr(pValues, cutoff)
```

**Arguments**

|         |   |
|---------|---|
| pValues | pValues to be used.   |
| cutoff  | The local fdr cutoff for rejection. Hypotheses with local fdr smaller then cutoff will be rejected. |

**Value**

A list containing:

|          |  |
|----------|--|
| locfdr   | Numeric vector with local FDR values for each case   |
| rejected | Logical vector indicating rejection/retention for each hypothesis when a cutoff is supplied. |

**Author(s)**

JonathanRosenblatt

**References**

- Strimmer, K. (2008). fdrtool: a versatile R package for estimating local and tail area-based false discovery rates. *Bioinformatics* 24: 1461-1462.
- Efron B., Tibshirani R., Storey J. D. and Tusher, V. (2001). Empirical Bayes Analysis of a Microarray Experiment. *Journal of the American Statistical Association* 96(456):1151-1160.

**Examples**

```
pvals<- runif(1000)^2
pval2locfdr(pvals)
pval2locfdr(pValues=pvals, cutoff=0.4)
```

---

pval2qval

*Strimmer et al.'s fdrtool-based q-values*

---

**Description**

The function pval2qval takes a vector of p-values and estimates for each case the tail area-based FDR, which can be regarded as a p-value corrected for multiplicity. This is done by calling the fdrtool function. If a cutoff is supplied, a vector of rejected hypotheses will be returned as well.

**Usage**

```
pval2qval(pValues, cutoff)
```

**Arguments**

|         |  |
|---------|--|
| pValues | Numeric vector of p-values to be used.   |
| cutoff  | The positive FDR cutoff for rejection. Hypotheses with qValues smaller then cutoff will be rejected. |

**Value**

A list containing:

- |          |  |
|----------|--|
| qValues  | A numeric vector with one q-value for each hypothesis.                                       |
| rejected | A logical vector indicating rejection/retention for each hypothesis when cutoff is supplied. |

**Author(s)**

JonathanRosenblatt

**References**

- Strimmer, K. (2008). fdrtool: a versatile R package for estimating local and tail area-based false discovery rates. *Bioinformatics* 24: 1461-1462.
- Storey, J. D. (2003) The Positive False Discovery Rate: A Bayesian Interpretation and the q-Value. *The Annals of Statistics* 31(6): 2013-2035

**Examples**

```
pvals<- runif(1000)^2
pval2qval(pvals)
pval2qval(pValues=pvals, cutoff=0.1)
```

---

pValuesPlot                    *A function plotting p-values...*

---

**Description**

A function plotting p-values

**Usage**

```
pValuesPlot(pValues)
```

**Arguments**

pValues                    A numeric containing the pValues to plot.

**Author(s)**

MarselScheer

---

Qvalue *Storey's (2001) q-value Procedure...*

---

### Description

Storey's (2001) q-value Procedure

### Usage

```
Qvalue(pValues, lambda=seq(0, 0.9, 0.05), pi0.method="smoother",
      fdr.level=NULL, robust=FALSE, smooth.df=3, smooth.log.pi0=FALSE,
      silent=FALSE)
```

### Arguments

|                |   |
|----------------|---|
| pValues        | pValues to be used (only necessary input)   |
| lambda         | Value of the tuning parameter to be used  |
| pi0.method     | Method for automatically choosing tuning parameter in the estimation of pi <sub>0</sub> . Either 'smoother' or 'bootstrap'  |
| fdr.level      | Level at which to control the FDR   |
| robust         | Logical, whether to make estimate more robust for small p-values.   |
| smooth.df      | Number of degrees of freedom to use when estimating pi <sub>0</sub> with the smoother.  |
| smooth.log.pi0 | Logical, if TRUE and pi0.method = 'smoother', pi <sub>0</sub> will be estimated by applying a smoother to a scatterplot of log(pi <sub>0</sub> ) estimates against the tuning parameter lambda. |
| silent         | logical scalar. If TRUE no output is generated.   |

### Details

The Qvalue procedure estimates the q-values for a given set of p-values. The q-value of a test measures the proportion of false positive incurred when that particular test is called significant. It gives the scientist a hypothesis testing error measure for each observed statistic with respect to the pFDR.

Note: If no options are selected, then the method used to estimate pi<sub>0</sub> is the smoother method described in Storey and Tibshirani (2003). The bootstrap method is described in Storey, Taylor and Siegmund (2004).

### Value

A list containing:

|              |  |
|--------------|--|
| qValues      | A vector of the estimated q-values   |
| pi0          | An estimate of the proportion of null hypotheses   |
| errorControl | A Mutoss S4 class of type errorControl, containing the type of error controlled by the function. |

**Author(s)**

HackNiklas

**References**

Storey, John (2001). The Positive False Discovery Rate: A Bayesian Interpretation and the Q-Value. The Annals of Statistics, Vol. 31, No. 6, 2013-2035.

**Examples**

```
pval <- c(runif(50), runif(50, 0, 0.01))
result <- Qvalue(pval)
result <- Qvalue(pval, lambda=0.5)
```

---

|               |  |
|---------------|--|
| ranktruncated | <i>Rank truncated p-Value procedure...</i> |
|---------------|--|

---

**Description**

Rank truncated p-Value procedure The program computes the exact distribution and with it the p-Value

**Usage**

```
ranktruncated(pValues, K, silent=FALSE)
```

**Arguments**

|         |   |
|---------|---|
| pValues | Vector of p-Values (not sorted)                       |
| K       | the number of hypotheses / p-Values being in w        |
| silent  | If true any output on the console will be suppressed. |

**Details**

This function computes the exact distribution of the product of at most  $K$  significant p-values of  $L > K$  observed p-values. Thus, one gets the pvalue from the exact distribution. This has certain advantages for genomewide association scans:  $K$  can be chosen on the basis of a hypothesised disease model, and is independent of sample size. Furthermore, the alternative hypothesis corresponds more closely to the experimental situation where all loci have fixed effects.

Please note that this method is implemented with factorials and binomial coefficients and the computation becomes numerical instable for large number of p-values.

**Value**

Used.pValue: List information about the used pValues; RTP: Test statistic and pValue

**Author(s)**

Frank Konietschke

**References**

Dubridge, F., Koeleman, B.P.C. (2003). Rank truncated product of P-values, with application to genomewide association scans. *Genet Epidemiol.* 2003 Dec;25(4):360-6

**Examples**

```
pvalues<-runif(10)
result <- ranktruncated(pvalues,K=2,silent=FALSE) # take the K=2 smallest pvalues
result <- ranktruncated(pvalues,K=2,silent=TRUE) # take the K=2 smallest pvalues
result <- ranktruncated(pvalues,K=5,silent=TRUE) # take the K=5 smallest pvalues
```

regwq

*REGWQ - Ryan / Einot and Gabriel / Welsch test procedure...***Description**

REGWQ - Ryan / Einot and Gabriel / Welsch test procedure This function computes REGWQ test for given data including  $p$  samples. It is based on a stepwise or layer approach to significance testing. Sample means are ordered from the smallest to the largest. The largest difference, which involves means that are  $r = p$  steps apart, is tested first at  $\alpha$  level of significance; if significant, means that are  $r < p$  steps apart are tested at a different  $\alpha$  level of significance and so on. Compare to the Student- Newman-Keuls test, the  $\alpha$  levels are adjusted for the  $p-1$  different layers by the formula  $\alpha_p = \alpha$ , if  $p=k$  or  $p=k-1$ ,  $\alpha_p = 1 - (1 - \alpha)^{p/k}$  otherwise. It might happen that the quantiles are not descending in  $p$ . In this case, they are adapted by  $c_k = \max_{2 \leq r \leq k} c_r$ ,  $k = 2, \dots, p$ . The REGWQ procedure, like Tukey's procedure, requires equal sample  $n$ 's. However, in this algorithm, the procedure is adapted to unequal sample sized which can lead to still conservative test decisions.

**Usage**

```
regwq(formula, data, alpha, MSE=NULL, df=NULL, silent=FALSE)
```

**Arguments**

|         |   |
|---------|---|
| formula | Formula defining the statistical model containing the response and the factors      |
| data    | dataset containing the response and the grouping factor                             |
| alpha   | The level at which the error should be controlled. By default it is $\alpha=0.05$ . |
| MSE     | Optional for a given variance of the data   |
| df      | Optional for a given degree of freedom  |
| silent  | If true any output on the console will be suppressed.                               |

**Value**

A list containing:

|                            |  |
|----------------------------|--|
| <code>adjPValues</code>    | A numeric vector containing the adjusted pValues   |
| <code>rejected</code>      | A logical vector indicating which hypotheses are rejected  |
| <code>statistics</code>    | A numeric vector containing the test-statistics  |
| <code>confIntervals</code> | A matrix containing only the estimates   |
| <code>errorControl</code>  | A Mutoss S4 class of type <code>errorControl</code> , containing the type of error controlled by the function. |

**Author(s)**

Frank Konietschke

**References**

Hochberg, Y. & Tamhane, A. C. (1987). Multiple Comparison Procedures, Wiley.

**Examples**

```
x = rnorm(50)
grp = c(rep(1:5,10))
dataframe <- data.frame(x,grp)
result <- regwq(x~grp, data=dataframe, alpha=0.05,MSE=NULL, df=NULL, silent = TRUE)
result <- regwq(x~grp, data=dataframe, alpha=0.05,MSE=NULL, df=NULL, silent = FALSE)
result <- regwq(x~grp, data=dataframe, alpha=0.05,MSE=1, df=Inf, silent = FALSE) # known variance
result <- regwq(x~grp, data=dataframe, alpha=0.05,MSE=1, df=1000, silent = FALSE) # known variance
```

---

|                     |               |
|---------------------|---------------|
| <code>reject</code> | <i>reject</i> |
|---------------------|---------------|

---

**Description**

Returns the highest rejected p-value and its index given some critical values.

**Usage**

```
reject(sorted, criticals)
```

**Arguments**

|                        |                 |
|------------------------|-----------------|
| <code>sorted</code>    | Sorted p-values |
| <code>criticals</code> | Critical values |

**Value**

A list with elements

|           |                                   |
|-----------|-----------------------------------|
| cutoff    | highest rejected p-value          |
| cut.index | index of highest rejected p-value |

---

|                |                                 |
|----------------|---------------------------------|
| requireLibrary | <i>Tries to load a package.</i> |
|----------------|---------------------------------|

---

**Description**

Tries to load a package. If this package does not exist, it will ask the user whether the package should be installed and loaded. If the user negates, we will raise an error via stop.

**Usage**

```
requireLibrary(package)
```

**Arguments**

|         |                 |
|---------|-----------------|
| package | Package to load |
|---------|-----------------|

**Value**

NULL

**Author(s)**

MuToss-Coding Team

---

|     |  |
|-----|--|
| rom | <i>Rom's (1990) step-up-procedure.</i> |
|-----|--|

---

**Description**

Rom's step-up-procedure is applied to pValues. The procedure controls the FWER in the strong sense if the pValues are stochastically independent.

**Usage**

```
rom(pValues, alpha, silent=FALSE)
```

**Arguments**

|         |  |
|---------|--|
| pValues | pValues to be used. They are assumed to be stochastically independent. |
| alpha   | the level at which the FWER shall be controlled.                       |
| silent  | if true any output on the console will be suppressed.                  |



**Details**

This function calculates the critical values by the formula given in Finner, H. and Roters, M. (2002) based on the joint distribution of order statistics. After that a step-up test is performed to reject hypotheses associated with pValues.

Since the formula for the critical values is recursive, the calculation of adjusted pValues is far from obvious and is not implemented here.

**Value**

A list containing:

|                |  |
|----------------|--|
| adjPValues     | A numeric vector containing the adjusted pValues   |
| rejected       | A logical vector indicating which hypotheses are rejected.   |
| criticalValues | A numeric vector containing critical values used in the step-up-down test.   |
| errorControl   | A Mutoss S4 class of type errorControl, containing the type of error controlled by the function and the level alpha. |

**Author(s)**

Marsel Scheer

**References**

Rom, D. M. (1990). A sequentially rejective test procedure based on a modified Bonferroni inequality. *Biometrika* 77, 663-665.

Finner, H. and Roters, M. (2002). Multiple hypotheses testing and expected type I errors. *Ann. Statist.* 30, 220-238.

**Examples**

```
r <- c(runif(50), runif(50, 0, 0.01))
result <- rom(r, 0.05)
result <- rom(r, 0.05, silent = TRUE)
```

---

SD

*A general step-down procedure.*

---

**Description**

A general step-down procedure.

**Usage**

```
SD(pValues, criticalValues)
```

**Arguments**

pValues            pValues to be used.  
 criticalValues    criticalValues for the step-down procedure

**Details**

Suppose we have  $n$  pValues and they are already sorted. The procedure starts with comparing pValues[1] with criticalValues[1]. If pValues[1]  $\leq$  criticalValues[1], then the hypothesis associated with pValues[1] is rejected and the algorithm carries on with second smallest pValue and criticalValue in the same way. The algorithm stops rejecting at the first index  $i$  for which pValues[ $i$ ] > criticalValues[ $i$ ]. Thus pValues[ $j$ ] is rejected if and only if pValues[ $i$ ]  $\leq$  criticalValues[ $i$ ] for all  $i \leq j$ .

**Value**

rejected logical vector indicating if hypotheses are rejected or retained.

**Author(s)**

MarselScheer

---

sidak

*Sidak correction*

---

**Description**

The classical Sidak correction returns adjusted p-values, ensuring strong FWER control under the assumption of independence of the input p-values. It only uses the fact that the probability of no incorrect rejection is the product over true nulls of those marginal probabilities (using the assumed independence of p-values).

**Usage**

```
sidak(pValues, alpha, silent=FALSE)
```

**Arguments**

pValues            pValues to be used.  
 alpha             The level at which the FWER shall be controlled (optional).  
 silent            logical scalar. If TRUE no output is generated.

**Details**

The procedure is more generally valid for positive orthant dependent test statistics.

It is recommended to use the step-down version of the Sidak correction instead (see SidakSD), which is valid under the exact same assumptions but is more powerful.

**Value**

A list containing:

|              |  |
|--------------|--|
| adjPValues   | A numeric vector containing the adjusted pValues   |
| rejected     | (if alpha is given) A logical vector indicating which hypotheses are rejected                    |
| errorControl | A Mutoss S4 class of type errorControl, containing the type of error controlled by the function. |

**Author(s)**

MuToss-Coding Team

**References**

Sidak, Z. (1967). Rectangular confidence regions for the means of multivariate normal distributions. *Journal of the American Statistical Association*, 62:626-633.

**Examples**

```
alpha <- 0.05
p <- c(runif(10, min=0, max=0.01), runif(10, min=0.9, max=1))
result <- sidak(p)
result <- sidak(p, alpha)
result <- sidak(p, alpha, silent=TRUE)
```

---

SidakSD

*Sidak-like (1987) step-down procedure*

---

**Description**

The Sidak-like (1987) step-down procedure is applied to pValues The Sidak-like step-down procedure is an improvement over the Holm's (1979) step-down procedure. The improvement is analogous to Sidak's correction over the original Bonferroni procedure. This Sidak-like step-down procedure assumes positive orthant dependent test statistics.

**Usage**

```
SidakSD(pValues, alpha, silent=FALSE)
```

**Arguments**

|         |   |
|---------|---|
| pValues | The used raw pValues.                                 |
| alpha   | The level at which the FWER shall be controlled.      |
| silent  | If true any output on the console will be suppressed. |

**Value**

A list containing:

|                             |  |
|-----------------------------|--|
| <code>adjPValues</code>     | a numeric vector containing the adjusted pValues   |
| <code>rejected</code>       | a logical vector indicating which hypotheses are rejected  |
| <code>criticalValues</code> | a numeric vector containing critical values used in the step-up-down test  |
| <code>errorControl</code>   | A Mutoss S4 class of type <code>errorControl</code> , containing the type of error controlled by the function and the level <code>alpha</code> . |

**Author(s)**

WerftWiebke

**References**

Hollander, B.S. and Copenhaver, M.D. (1987). An Improved Sequentially Rejective Bonferroni Test Procedure. *Biometrics*, 43(2):417-423, 1987.

**Examples**

```
alpha <- 0.05
p <- c(runif(10, min=0, max=0.01), runif(10, min=0.9, max=1))
result <- SidakSD(p, alpha)
result <- SidakSD(p, alpha, silent=TRUE)
```

---

simulation

*Simulation studies*

---

**Description**

This function generates data according to a specified function and parameters and then applies specified procedures to the generated data. The generated data is stored in `$data` and the results are stored in `$results`. In order to recognize which results and generated data belong together every result contains an element `$data.set.number`. The data generating function must return a list which contains a list `$procInput`. Every element in `$procInput` will be used as an input parameter for the procedures provided by `listOfProcedures`.

**Usage**

```
simulation(replications, DataGen, listOfProcedures, discardProcInput=FALSE)
```

**Arguments**

|                  |  |
|------------------|--|
| replications     | The number of replications. This means how many simulation runs will be performed.   |
| DataGen          | A list that contains the function and parameters for generating data which will be analyzed by the procedures in listOfProcedures. It must contain the elements \$funName (character) \$fun (function) |
| listOfProcedures | A list of lists which contains the procedures with their parameters for the simulation.  |
| discardProcInput | A list of lists which contains the procedures with their parameters  |

**Value**

A list with 2 elements. \$data contains the objects generated by DataGen. \$results contains the objects generated by the procedures augmented by the data set number and the parameter constellation.

**Author(s)**

MarselScheer

**Examples**

```
# this function generates pValues
myGen <- function(n, n0) {
  list(procInput=list(pValues = c(runif(n-n0, 0, 0.01), runif(n0))),
    groundTruth = c(rep(FALSE, times=n-n0), rep(TRUE, times=n0)))
}

sim <- simulation(replications = 10, list(funName="myGen", fun=myGen, n=200, n0=c(50,100)),
  list(list(funName="BH",
    fun=function(pValues, alpha) BH(pValues, alpha, silent=TRUE), alpha=c(0.25, 0.5)),
  list(funName="holm", fun=holm, alpha=c(0.25, 0.5), silent=TRUE)))

# the following happend:
# Call myGen(200,50) and append result to sim$data
# Apply bonferroni and holm each with alpha 0.25 and 0.5 to this data set
# Append the results to sim$results
# Repeat this 10 times.
# Call myGen(200, 100) and append result to sim$data
# Apply bonferroni and holm each with alpha 0.25 and 0.5 to this data set
# Append the results to sim$results
# Repeat this 10 times.

length(sim$data)
length(sim$results)

# we now reproduce the 6th item in results
print(sim$results[[6]]$data.set.number)
```

```

print(sim$results[[6]]$parameters)

all(BH(sim$data[[2]]$procInput$pValues, 0.5, silent=TRUE)$adjPValues == sim$results[[6]]$adjPValues)

#
# Just calculating some statistics and making some plots
NumberOfType1Error <- function(data, result) sum(data$groundTruth * result$rejected)
result.all <- gatherStatistics(sim, list(NumOfType1Err = NumberOfType1Error))
result <- gatherStatistics(sim, list(NumOfType1Err = NumberOfType1Error),
  list(median=median, mean=mean, sd=sd))
print(result)
require(lattice)
histogram(~NumOfType1Err | method*alpha, data = result.all$statisticDF)
barchart(NumOfType1Err.median + NumOfType1Err.mean ~ method | alpha, data = result$statisticDF)

```

snk

*Student - Newman - Keuls rejective test procedure.***Description**

Student - Newman - Keuls rejective test procedure. The procedure controls the FWER in the WEAK sense.

**Usage**

```

snk(formula, data, alpha, MSE=NULL, df=NULL, silent=FALSE)
snk.wrapper(model, data, alpha, silent=FALSE)

```

**Arguments**

|         |   |
|---------|---|
| formula | Formula defining the statistical model containing the response and the factor levels. |
| model   | Model with formula, containing the response and the factor levels                     |
| data    | dataset containing the response and the grouping factor.                              |
| alpha   | The level at which the error should be controlled. By default it is alpha=0.05.       |
| MSE     | Optional for a given variance of the data.  |
| df      | Optional for a given degree of freedom.   |
| silent  | If true any output on the console will be suppressed.                                 |

**Details**

This function computes the Student-Newman-Keuls test for given data including  $p$  samples. The Newman-Keuls procedure is based on a stepwise or layer approach to significance testing. Sample means are ordered from the smallest to the largest. The largest difference, which involves means that are  $r = p$  steps apart, is tested first at  $\alpha$  level of significance; if significant, means that are  $r = p - 1$  steps apart are tested at  $\alpha$  level of significance and so on. The Newman-Keuls procedure provides an  $r$ -mean significance level equal to  $\alpha$  for each group of  $r$  ordered means, that is, the probability of

falsely rejecting the hypothesis that all means in an ordered group are equal to  $\alpha$ . It follows that the concept of error rate applies neither on an experimentwise nor on a per comparison basis-the actual error rate falls somewhere between the two. The Newman-Keuls procedure, like Tukey's procedure, requires equal sample  $n$ 's. However, in this algorithm, the procedure is adapted to unequal sample sized which can lead to still conservative test decisions.

It should be noted that the Newman-Keuls and Tukey procedures require the same critical difference for the first comparison that is tested. The Tukey procedure uses this critical difference for all the remaining tests, whereas the Newman-Keuls procedure reduces the size of the critical difference, depending on the number of steps separating the ordered means. As a result, the Newman-Keuls test is more powerful than Tukey's test. Remember, however, that the Newman-Keuls procedure does not control the experimentwise error rate at  $\alpha$ .

## Value

A list containing:

|               |  |
|---------------|--|
| adjPValues    | A numeric vector containing the adjusted pValues   |
| rejected      | A logical vector indicating which hypotheses are rejected  |
| statistics    | A numeric vector containing the test-statistics  |
| confIntervals | A matrix containing only the estimates   |
| errorControl  | A Mutoss S4 class of type errorControl, containing the type of error controlled by the function. |

## Author(s)

Frank Konietschke

## References

Keuls M (1952). "The use of the studentized range in connection with an analysis of variance". *Euphytica* 1: 112-122

## Examples

```
x = rnorm(50)
grp = c(rep(1:5,10))
dataframe <- data.frame(x,grp)
result <- snk(x~grp, data=dataframe, alpha=0.05,MSE=NULL, df=NULL, silent = TRUE)
result <- snk(x~grp, data=dataframe,alpha=0.05,MSE=NULL, df=NULL, silent = FALSE)
result <- snk(x~grp, data=dataframe,alpha=0.05,MSE=1, df=Inf, silent = FALSE) # known variance
result <- snk(x~grp, data=dataframe,alpha=0.05,MSE=1, df=1000, silent = FALSE) # known variance
```

---

`storey_pi0_est`*Storey-Taylor-Siegmund estimation of  $\pi_0$  (finite sample version)*

---

**Description**

The Storey-Taylor-Siegmund procedure for estimating  $\pi_0$  is applied to pValues. The formula is equivalent to that in Schweder and Spjøtvoll (1982), page 497, except the additional '+1' in the nominator that introduces a conservative bias which is proven to be sufficiently large for FDR control in finite families of hypotheses if the estimation is used for adjusting the nominal level of a linear step-up test.

**Usage**

```
storey_pi0_est(pValues, lambda)
```

**Arguments**

|         |   |
|---------|---|
| pValues | The raw p-values for the marginal test problems |
| lambda  | A tuning parameter in the interval (0, 1)       |

**Value**

A list containing:

|        |   |
|--------|---|
| pi0    | A numeric number containing the estimated value of $\pi_0$          |
| lambda | A numeric number containing the tuning parameter for the estimation |

**Author(s)**

MarselScheer

**References**

Schweder, T. and Spjøtvoll, E. (1982). Plots of P-values to evaluate many tests simultaneously. *Biometrika* 69, 3, 493-502.

Storey, J. D., Taylor, J. E. and Siegmund, D. (2004). Strong control, conservative point estimation and simultaneous conservative consistency of false discovery rates: a unified approach. *JRSS B* 66, 1, 187-205.

**Examples**

```
my.pvals <- c(runif(50), runif(50, 0, 0.01))
result <- storey_pi0_est(my.pvals, 0.5)
```



---

SU *A general step-up procedure.*

---

**Description**

A general step-up procedure.

**Usage**

SU(pValues, criticalValues)

**Arguments**

pValues            pValues to be used.  
criticalValues    criticalValues for the step-up procedure

**Details**

Suppose we have  $n$  pValues and they are already sorted. The procedure starts with comparing pValues[n] with criticalValues[n]. If pValues[n] > criticalValues[n], then the hypothesis associated with pValues[n] is retained and the algorithm carries on with the next pValue and criticalValue, here for example pValues[n-1] and criticalValues[n-1]. The algorithm stops retaining at the first index  $i$  for which pValues[i] <= criticalValues[i]. Thus pValues[j] is rejected if and only if there exists an index  $i$  with  $j < i$  and pValues[i] <= criticalValues[i].

**Value**

rejected logical vector indicating if hypotheses are rejected or retained.

**Author(s)**

MarselScheer

---

SUD *A general step-up-down procedure.*

---

**Description**

A general step-up-down procedure.

**Usage**

SUD(pValues, criticalValues, startIDX\_SUD)

**Arguments**

|                |  |
|----------------|--|
| pValues        | pValues to be used.  |
| criticalValues | criticalValues for the step-up-down procedure  |
| startIDX_SUD   | the index (between 1 and length(pValues)) used for the first comparison of pValues[startIDX_SUD] and criticalValues[startIDX_SUD]. Depending on the result of this comparison the algorithm decides to proceed in step-up or step-down manner. |

**Details**

Suppose we have  $n$  pValues and they are already sorted. The procedure compares pValues[startIDX\_SUD] with criticalValues[startIDX\_SUD] and then proceeds in step-up or step-down manner, depending on the result of this initial comparison.

If pValues[startIDX\_SUD]  $\leq$  criticalValues[startIDX\_SUD], then the procedure rejects the hypotheses associated with pValues[1], ..., pValues[startIDX\_SUD] and carries on in a step-down manner from startIDX\_SUD to  $n$  to reject additional hypotheses.

If pValues[startIDX\_SUD]  $>$  criticalValues[startIDX\_SUD], then the procedure retains hypotheses associated with pValues[startIDX\_SUD], ..., pValues[ $n$ ] and carries on in a step-up manner with pValues[startIDX\_SUD - 1], ..., pValues[1].

If startIDX\_SUD equals  $n$  the algorithm behaves like a step-up procedure.

If startIDX\_SUD equals 1 the algorithm behaves like a step-down procedure.

**Value**

rejected logical vector indicating if hypotheses are rejected or retained.

**Author(s)**

MuToss-Coding Team

---

TSBKY\_pi0\_est

*Two-step estimation method of Benjamini, Krieger and Yekutieli for estimating  $\pi_0$*

---

**Description**

The two-step estimation method of Benjamini, Krieger and Yekutieli for estimating  $\pi_0$  is applied to pValues. It consists of the following two steps: Step 1. Use the linear step-up procedure at level  $\alpha' = \alpha/(1+\alpha)$ . Let  $r_1$  be the number of rejected hypotheses. If  $r_1=0$  do not reject any hypothesis and stop; if  $r_1=m$  reject all  $m$  hypotheses and stop; otherwise continue. Step 2. Let  $\hat{m}_0 = (m - r_1)$  and  $\hat{\pi}_0 = \hat{m}_0/m$ .

**Usage**

TSBKY\_pi0\_est(pValues, alpha)

**Arguments**

pValues            The raw p-values for the marginal test problems  
 alpha             The parameter (to be interpreted as significance level) for the procedure

**Value**

pi0.TSBKY        The estimated proportion of true null hypotheses.

**Author(s)**

WerftWiebke

**References**

Benjamini, Y., Krieger, A. and Yekutieli, D. (2006). Adaptive linear step-up procedures that control the false discovery rate *Biometrika* 93, 3, page 495.

**Examples**

```
my.pvals <- c(runif(50), runif(50, 0, 0.01))
result <- TSBKY_pi0_est(my.pvals, 0.1)
```

---

|               |  |
|---------------|--|
| tukey.wrapper | <i>Tukey HSD test and simultaneous confidence intervals for all pairs comparisons...</i> |
|---------------|--|

---

**Description**

Tukey HSD test and simultaneous confidence intervals for all pairs comparisons in factorial designs. The procedure controls the FWER in the strong sense.

**Usage**

```
tukey.wrapper(model, alpha, factorC)
```

**Arguments**

model            A fitted model, for example an object returned by lm, glm, or aov etc. It is assumed that coef and vcov methods are available for model. Usually, it is an aov fit  
 alpha            The significance level  
 factorC         Specifies a factor

**Details**

this function, it is possible to compute all pairs comparisons for expectations and simultaneous confidence intervals in factorial linear models. Hereby, the all-pairs comparisons can be performed for user given effects. The overall variance is estimated by the linear model as well as the degree of freedom used by the studentized range distribution.

**Value**

A list containing:

`adjPValues` A numeric vector containing the adjusted pValues  
`rejected` A logical vector indicating which hypotheses are rejected  
`confIntervals` A matrix containing the estimates and the lower and upper confidence bound  
`errorControl` A Mutoss S4 class of type `errorControl`, containing the type of error controlled by the function.

**Author(s)**

Frank Konietschke et al.

**Examples**

```
data(warpbreaks)
# Tukey contrast on the levels of the factor "Tension"

tukey.wrapper(aov(breaks ~ tension, data = warpbreaks), factorC="tension",alpha=0.05)

# Two-way anova with interaction
tukey.wrapper(aov(breaks ~ tension*wool, data = warpbreaks),alpha=0.05,factorC="tension")
# Two-way anova without interaction

tukey.wrapper(aov(breaks ~ tension+wool, data = warpbreaks),alpha=0.05,factorC="tension")
tukey.wrapper(aov(breaks ~ tension, data = warpbreaks),alpha=0.05,factorC="tension")

data(iris)
tukey.wrapper(lm(Sepal.Length ~ Species, data=iris),alpha=0.05, factorC="Species")
```

---

|           |  |
|-----------|--|
| two.stage | <i>A p-value procedure which controls the FDR for independent test statistics.</i> |
|-----------|--|

---

**Description**

A p-value procedure which controls the FDR for independent test statistics.

**Usage**

```
two.stage(pValues, alpha)
```

**Arguments**

|         |                                |
|---------|--------------------------------|
| pValues | A numeric vector of p-values.  |
| alpha   | The FDR error rate to control. |

**Details**

In the Benjamini-Krieger-Yekutieli two-stage procedure the linear step-up procedure is used in stage one to estimate  $m_0$  which is re-plugged in a linear step-up. This procedure is more powerful than non-adaptive procedures, while still controlling the FDR. On the other hand, error control is not guaranteed under dependence in which case more conservative procedures should be used (e.g. BH).

**Value**

A list containing:

|                |  |
|----------------|--|
| rejected       | A logical vector indicating which hypotheses are rejected  |
| criticalValues | A numeric vector containing critical values used in the step-up-down test.   |
| adjPValues     | A numeric vector containing adjusted p-values.   |
| pi0            | An estimate of the proportion of true null hypotheses among all hypotheses ( $\pi_0 = m_0/m$ ).  |
| errorControl   | A Mutoss S4 class of type <code>errorControl</code> , containing the type of error controlled by the function and the level <code>alpha</code> . |

**Author(s)**

JonathanRosenblatt

**Examples**

```
pvals<- runif(100)^2
two.stage(pvals, 0.1)
```

---

|                  |                                 |
|------------------|---------------------------------|
| twosamp.marginal | <i>Marginal two sample test</i> |
|------------------|---------------------------------|

---

**Description**

The robust version uses the Wilcoxon-Mann-Whitney test, otherwise a two-sample t-test will be performed.

**Usage**

```
twosamp.marginal(data, model, robust, alternative, psi0, equalvar)
```

**Arguments**

|             |   |
|-------------|---|
| data        | the data set  |
| model       | the result of a call to <code>twosamp.model(classlabel)</code>  |
| robust      | a logical variable indicating whether a two sample t-test or a Wilcoxon-Mann-Whitney test should be used.                                   |
| alternative | a character string specifying the alternative hypothesis, must be one of <code>two.sided</code> , <code>greater</code> or <code>less</code> |
| psi0        | a numeric that defines the hypothesized null value  |
| equalvar    | a logical variable indicating whether to treat the two variances as being equal   |

**Details**

A vector of classlabels needs to be provided to distinguish the two groups.

**Value**

pValues            A numeric vector containing the unadjusted pValues

**Author(s)**

MuToss-Coding Team

**References**

Wilcoxon, F. (1945). Individual Comparisons by Ranking Methods. *Biometrics Bulletin* 1:80-83.  
 Mann, H. and Whitney, D. (1947). On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics* 18:50-60  
 Student (1908). The probable error of a mean. *Biometrika*, 6(1):1-25.

---

twostageBR

*Blanchard-Roquain (2009) 2-stage adaptive step-up...*

---

**Description**

Blanchard-Roquain (2009) 2-stage adaptive step-up

**Usage**

```
twostageBR(pValues, alpha, lambda=1, silent=FALSE)
```

**Arguments**

|         |  |
|---------|--|
| pValues | the used p-values (assumed to be independent)                                |
| alpha   | the level at which the FDR should be controlled.                             |
| lambda  | parameter of the procedure, should belong to (0, 1/alpha) (lambda=1 default) |
| silent  | if true any output on the console will be suppressed.                        |

**Details**

This is an adaptive linear step-up procedure where the proportion of true nulls is estimated using the Blanchard-Roquain 1-stage procedure with parameter lambda, via the formula

estimated  $\pi_0 = (m - R(\alpha, \lambda) + 1) / (m * (1 - \lambda * \alpha))$

where  $R(\alpha, \lambda)$  is the number of hypotheses rejected by the BR 1-stage procedure, alpha is the level at which FDR should be controlled and lambda an arbitrary parameter belonging to (0, 1/alpha) with default value 1. This procedure controls FDR at the desired level when the p-values are independent.

**Value**

A list containing:

|              |  |
|--------------|--|
| rejected     | A logical vector indicating which hypotheses are rejected  |
| errorControl | A Mutoss S4 class of type errorControl, containing the type of error controlled by the function and the level alpha. |

**Author(s)**

GillesBlanchard

**References**

Blanchard, G. and Roquain, E. (2009) Adaptive False Discovery Rate Control under Independence and Dependence Journal of Machine Learning Research 10:2837-2871.

# Index

- \* **datasets**
  - notterman, 43
- ABH\_pi0\_est, 3
- adaptiveBH, 4
- adaptiveSTS, 5
- adjPValuesPlot, 6
- aorc, 7
- augmentation, 8
  
- BH, 9
- BL, 11
- BlaRoq, 12
- bonferroni, 13
- BR\_pi0\_est, 14
- BY, 15
  
- calculateBetaAdjustment, 16
- compareMutoss, 17
  
- ErrorControl-class, 19
  
- fisher22.marginal, 19
- fisher22.model (mutoss.models), 41
- fisher22\_fast, 20
- fisher23.marginal, 21
- fisher23.model (mutoss.models), 41
- fisher23\_fast, 21
- ftest.marginal, 22
- ftest.model (mutoss.models), 41
  
- gao, 23
- gatherParameters, 24
- gatherStatistics, 25
  
- hochberg, 27
- holm, 28
- hommel, 29
  
- indepBR, 30
  
- jointCDF.orderedUnif, 31
  
- linearStepUp, 32
  
- mu.compare.adjusted (compareMutoss), 17
- mu.compare.critical (compareMutoss), 17
- mu.compare.summary (compareMutoss), 17
- mu.test.class, 33
- mu.test.name, 34
- mu.test.rates, 34
- mu.test.same.data, 35
- mu.test.type, 36
- multcomp.wrapper, 36
- multiple.down, 38
- multiple.down.adjust, 39
- Mutoss-class, 39
- mutoss.apply, 40
- mutoss.models, 41
- mutoss.plotCI, 42
- MutossMethod (MutossMethod-class), 42
- MutossMethod-class, 42
  
- notterman, 43
- nparcomp, 43
- nparcomp.wrapper, 45
  
- onesamp.marginal, 46
- onesamp.model (mutoss.models), 41
- oracleBH, 47
  
- paired.marginal, 48
- paired.model (mutoss.models), 41
- printRejected, 49
- pval2locfdr, 49
- pval2qval, 50
- pValuesPlot, 51
  
- Qvalue, 52
  
- ranktruncated, 53
- regwq, 54
- reject, 55
- requireLibrary, 56



rom, [56](#)

SD, [57](#)

sidak, [58](#)

SidakSD, [59](#)

simulation, [60](#)

snk, [62](#)

storey\_pi0\_est, [64](#)

SU, [65](#)

SUD, [65](#)

T.Test.tumor.vs.normal (notterman), [43](#)

TSBKY\_pi0\_est, [66](#)

tukey.wrapper, [67](#)

two.stage, [68](#)

twosamp.marginal, [69](#)

twosamp.model (mutoss.models), [41](#)

twostageBR, [70](#)