Package 'metagen'

February 20, 2015

Version 1.0

Date 2014-05

Title Inference in Meta Analysis and Meta Regression

Description Provides methods for making inference in the random effects meta regression model such as point estimates and confidence intervals for the heterogeneity parameter and the regression coefficients vector. Inference methods are based on different approaches to statistical inference. Methods from three different schools are included: methods based on the method of moments approach, methods based on likelihood, and methods based on generalised inference. The package also includes tools to run extensive simulation studies in parallel on high performance clusters in a modular way. This allows extensive testing of custom inferential methods with all implemented state-of-the-art methods in a standardised way. Tools for evaluating the performance of both point and interval estimates are provided. Also, a large collection of different pre-defined plotting functions is implemented in a ready-to-use fashion.

Author Thomas W. D. Möbius <kontakt@thomasmoebius.de>

Maintainer Thomas W. D. Möbius <kontakt@thomasmoebius.de>

URL http://00tau.github.io/metagen/

License GPL-3

Imports MASS, lhs, plyr, BBmisc, ParamHelpers, BatchJobs, BatchExperiments, ggplot2, metafor

LazyData yes

ByteCompile yes

Encoding UTF-8

Collate 'example-bcg.R' 'methods-for-batch-distribution.R' 'methods-for-data-generation.R' 'methods-for-experiments.R' 'methods-for-inference.R' 'plot-functions.R' 'plot-interval-estimates.R' 'plot-pivotal-distributions.R' 'plot-point-estimates.R' 'zzz.R'

NeedsCompilation no

Repository CRAN

Date/Publication 2014-05-27 15:43:54

R topics documented:

`bcgVaccineData` *Example: Setting up the BCG-data set*

Description

Exemplary data set of 14 clinical trials evaluating BCG vaccine efficacy.

Usage

```
bcgVaccineData(sgnf = 0.05)
```

Arguments

`sgnf`	significance level of the confidence intervals for the relative risks.

Details

Reads in the BCG vaccine efficacy data from the metafor package and adds some statistics to the data such as the log-relative risk, study size, measurements of balance, confidence intervals of the responses, and the like.

Value

Returns a data set of 13 clinical trials which evaluated the efficacy of the BCG vaccine. The data set is an exact copy of the data set found in the dat.bcg data frame provided by the metafor package.

Examples

```
bcgVaccineData()
```

---

boxBias *Plotting performance: Box plots for bias*

Description

Box plots for the bias.

Usage

```
boxBias(res)
```

Arguments

res	The collected results from a computer experiment.

Value

A plot object.

---

boxByConfidence *Plotting performance: Box plots for target value confidence-coverage*

Description

Plotting performance: Box plots for target value confidence-coverage

Usage

```
boxByConfidence(res)
```

Arguments

res	The collected results from a computer experiment.

Value

A plot object.

---

boxByMethod *Plotting performance: Box plots for target value confidence-coverage*

Description

Plotting performance: Box plots for target value confidence-coverage

Usage

```
boxByMethod(res)
```

Arguments

res	The collected results from a computer experiment.

Value

A plot object.

---

boxByType *Plotting performance: Box plots for target value confidence-coverage*

Description

Plotting performance: Box plots for target value confidence-coverage

Usage

```
boxByType(res)
```

Arguments

res	The collected results from a computer experiment.

Value

A plot object.

boxMSE *Plotting performance: Box plots for mean squared error*

Description

Box plots for mean squared error.

Usage

```
boxMSE(res)
```

Arguments

res	The collected results from a computer experiment.

Value

A plot object.

boxSD *Plotting performance: Box plots for standard deviation*

Description

Box plots for standard deviation.

Usage

```
boxSD(res)
```

Arguments

res	The collected results from a computer experiment.

Value

A plot object.

cbbPalette *Colour palettes for colour blind people*

Description

The palette with black.

Usage

```
cbbPalette
```

Format

chr [1:8] "#000000" "#E69F00" "#56B4E9" "#009E73" "#F0E442" ...

Details

This palette is directly taken from

http://www.cookbook-r.com/Graphs/Colors_(ggplot2)/

Hence, I don't take any credit for this.

Examples

```
scale_fill_discrete <- function(...) scale_fill_manual(...,
  values=cbbPalette)
scale_colour_discrete <- function(...) scale_fill_manual(...,
  values=cbbPalette)
```

cbgPalette *Colour palettes for colour blind people*

Description

The palette with grey.

Usage

```
cbgPalette
```

Format

chr [1:8] "#999999" "#E69F00" "#56B4E9" "#009E73" "#F0E442" ...

Details

This palette is directly taken from

http://www.cookbook-r.com/Graphs/Colors_(ggplot2)/

Hence, I don't take any credit for this.

Examples

```
scale_fill_discrete <- function(...) scale_fill_manual(...,
  values=cbgPalette)
scale_colour_discrete <- function(...) scale_fill_manual(...,
  values=cbgPalette)
```

---

collectAllExperiments *Running a computer experiment – Collect all the results*

Description

Collects all the results of all finished experiments in the given registry for all predefined algorithms.

Usage

```
collectAllExperiments(reg)
```

Arguments

reg	A valid registry generated by 'makeExperimentRegistry'.

Value

List of data frames containing the performance measures of all point and interval estimates for the heterogeneity and the regression coefficients.

---

collectExperiments *Running a computer experiment – Collect specific results*

Description

Collects specific results of all finished experiments in the given registry for a given pattern.

Usage

```
collectExperiments(reg, pattern)
```

Arguments

reg	A valid registry generated by 'makeExperimentRegistry'.
pattern	string containing the algorithm pattern for which the collection shall be performed.

Value

List of data frames containing the performance measures of all point and interval estimates for the heterogeneity and the regression coefficients.

---

designB *Design: Binomial responses*

Description

Method for generating a sampling design for data generation following a binomial-Gaussian model.

Usage

```
designB(n, h_bounds, a_bounds, s_bounds, r, x)
```

Arguments

n	resolution of the heterogeneity. n is the number of of different heterogeneity parameters in the design.
h_bounds	bounds of the heterogeneity.
a_bounds	bounds of the balancing factor of group assignments.
s_bounds	bounds of the study sizes.
r	fixed risk in the control.
x	design matrix.

Details

Generates a sampling design for the heterogeneity 'h', balancing factors 'a1', ..., 'ak' of group assignments, and study sizes 's1', ..., 'sk'. This design can be used for testing methods for inference for the random effects meta regression model since the logarithm of relative risks of each study is approximately Gaussian distributed. One may use methods that adjust for uncertainty in the heteroscedasticity estimates by additionally considering the size of the respected studies.

Points in the design are selected via a maxi-min hypercube sampling using the 'lhs' package in a predefined parameter cube.

Value

Function returns a data frame. Each line of this data frame can be an input to the function 'rB' which is used to sample data from such a design.

Examples

```
dB <- designB(n=15L, h_bounds=c(0,1), a_bounds=c(-.3,3),
  s_bounds=c(200L,2000L), r=0.03, x=cbind(1,1:5))

if(!all(dim(dB) == c(15,2*dim(cbind(1,1:5))[1]+2))) {
  stop("Wrong dimension")
}
```

`designD` *Design: Gaussian responses (unknown heteroscedasticity)*

Description

Method for generating a sampling design for data generation following a random effects meta regression model with unknown heteroscedasticity.

Usage

```
designD(n, h_bounds, d_bounds, s_bounds, x)
```

Arguments

`n`	resolution of the heterogeneity and heteroscedasticity parameters, i.e., the number of of different (heterogeneity, heteroscedasticity, sizes) tuple in the design.
`h_bounds`	bounds of the heterogeneity.
`d_bounds`	bounds of the heteroscedasticity.
`s_bounds`	bounds of the study sizes.
`x`	design matrix.

Details

Generates a sampling design for the heterogeneity 'h', heteroscedasticity 'd1', ..., 'dk', and study sizes 's1', ..., 'sk'. This design can be used for testing methods that adjust for uncertainty in the heteroscedasticity estimates by additionally considering the size of the respected studies.

Points in the design are selected via a maxi-min hypercube sampling using the 'lhs' package in a predefined parameter cube.

Value

Function returns a data frame. Each line of this data frame can be an input to the function 'rD' which is used to sample data from such a design.

Examples

```
dD <- designD(n=15L, h_bounds=c(0,1), d_bounds=c(0.01,2),
  s_bounds=c(200L,2000L), x=cbind(1,1:7))

if(!all(dim(dD) == c(15,2*dim(cbind(1,1:7))[1]+1))) {
  stop("Wrong dimension")
}
```

`designY` *Design: Gaussian responses (known heteroscedasticity)*

Description

Method for generating a sampling design for data generation following a random effects meta regression model with known heteroscedasticity.

Usage

```
designY(n, h_bounds, d_bounds, x)
```

Arguments

n	resolution of the heterogeneity and heteroscedasticity parameters, i.e. the number of of different (heterogeneity, heteroscedasticity) pairs in the design.
h_bounds	bounds of the heterogeneity.
d_bounds	bounds of the heteroscedasticity.
x	design matrix.

Details

Generates a sampling design for the heterogeneity 'h' and a heteroscedasticity 'd1', ..., 'dk'.

Points in the design are selected via a maxi-min hypercube sampling using the 'lhs' package in a predefined parameter cube.

Value

Function returns a data frame. Each line of this data frame can be an input to the function 'rY' which is used to sample data from such a design.

Examples

```
dY <- designY(n=15L, h_bounds=c(0,1), d_bounds=c(0.01,2),
x=cbind(1,1:7))

if(!all(dim(dY) == c(15,dim(cbind(1,1:7))[1]+1))) {
  stop("Wrong dimension")
}
```

dvec *Data generation: Sampling data of clinical trials*

Description

Calculates the variance estimate of log risk ratios from a study in the right format. See the example below for details.

Usage

```
dvec(study)
```

Arguments

study	Study data of a clinical trial with binomial outcomes.

Examples

```
h_test <- .03
x_test <- cbind(1,1:13)
b_test <- c(0.02, 0.03)
s_test <- rep(2000, 13)
a_test <- rep(.3, 13)
rBinomGauss(  h=h_test, s=s_test, a=a_test, r=0.03
            , x=x_test, b=b_test)$study -> test
yvec(test)
dvec(test)
```

experimentD *Running a computer experiment*

Description

Runs a computer experiment that evaluates the performance of different inference methods for the random effects meta regression model with respect to heterogeneity and regression coefficients.

Usage

```
experimentD(n, h, d, s, x, b, sgnf, piv_draws)
```

Arguments

n	number of draws.
h	heterogeneity.
d	heteroscedasticity.
s	vector study sizes.
x	design matrix.
b	regression coefficients.
sgnf	significance levels.
piv_draws	privotal draws.

Details

This also includes methods adjusting for uncertainty in the heteroscedasticity vector. In particular, the study sizes need to be known, here.

Value

Data frame of accumulated performance measures.

Examples

```
h_test <- 0.03
x_test <- cbind(1,1:7)
b_test <- c(.5, .25)
sgnf_test <- c(0.025, 0.01)

set.seed(5133568) # for reproducibility
d_test <- rchisq(7, df=0.02)
s_test <- runif(7, min=200, max=2000)

# In an actual computer experiment, use 'piv_draws=1000' instead!!
experimentD(n=5, h=h_test, d=d_test, s=s_test, x=x_test, b=b_test,
  sgnf=sgnf_test, piv_draws=50)
```

---

experimentY *Running a computer experiment*

Description

Runs a computer experiment that evaluates the performance of different inference methods for the random effects meta regression model with respect to heterogeneity and regression coefficients.

Usage

```
experimentY(n, h, d, x, b, sgnf, piv_draws)
```

Arguments

n	number of draws.
h	heterogeneity.
d	heteroscedasticity.
x	design matrix.
b	regression coefficients.
sgnf	significance levels.
piv_draws	privotal draws.

Value

Data frame of accumulated performance results.

Examples

```
h_test <- 0.03
x_test <- cbind(1,1:7)
b_test <- c(.5, .25)
sgnf_test <- c(0.025, 0.01)

set.seed(5133568) # for reproducibility
d_test <- rchisq(7, df=0.02)

# In an actual computer experiment, use 'piv_draws=1000' instead!!
experimentY(n=5, h=h_test, d=d_test, x=x_test, b=b_test,
  sgnf=sgnf_test, piv_draws=50)
```

---

formulaL *Regression coefficients: formulaL*

Description

Calculate pivotal quantities for the regression coefficients using the method: formulaL form the dissertation.

Usage

```
formulaL(y, d, h, g, x)
```

Arguments

y	k-vector of responses.
d	k-vector of heteroscedasticity.
h	scalar of heterogeneity.
g	p-vector of some p-variate Gaussian draw.
x	design k-p-matrix.

Details

Algorithm for calculating a single generalised pivotal quantity for the regression coefficients for given generalised pivotal quantities for the heterogeneity using the univariate version of the pivotal formula.

Value

A p-vector.

Examples

```
bcg   <- bcgVaccineData()
bcg_y <- bcg$logrisk
bcg_d <- bcg$sdiv
bcg_x <- cbind(1,bcg$x)

# When for example using the Mandel-Paule estimate:
bcg_h <- pfunc(y=bcg_y, d=bcg_d, x=bcg_x)(dim(bcg_x)[1] -
  dim(bcg_x)[2])

set.seed(51351) # for reproducibility
random_g <- rnorm(dim(bcg_x)[2])
formulaL(y=bcg_y, d=bcg_d, h=bcg_h, g=random_g, x=bcg_x)

# The function can also be used when planing to perform
# a meta regression with no intercept, and only a singel
# covariate (i.e. dim(x) = 1).  In this case,
# the design matrix can simply be provided by a vector.
set.seed(51351) # for reproducibility
random_g <- rnorm(1)
formulaL(y=bcg_y, d=bcg_d, h=bcg_h, g=random_g, x=bcg$x)

# When performing a meta analysis, provide the function
# with a vector of 1s.
formulaL(y=bcg_y, d=bcg_d, h=bcg_h, g=random_g, x=rep(1,
  length(bcg_y)))
```

`formulaR` *Regression coefficients: formulaR*

Description

Calculate pivotal quantities for the regression coefficients using the method: formulaR form the dissertation.

Usage

```
formulaR(y, d, h, g, x)
```

Arguments

y	k-vector of responses.
d	k-vector of heteroscedasticity.
h	scalar of heterogeneity.
g	p-vector of some p-variate Gaussian draw.
x	design k-p-matrix.

Details

Algorithm for calculating a single generalised pivotal quantity for the regression coefficients for given generalised pivotal quantities for the heterogeneity using the multivariate version of the pivotal formula.

Value

A p-vector.

Examples

```
bcg   <- bcgVaccineData()
bcg_y <- bcg$logrisk
bcg_d <- bcg$sdiv
bcg_x <- cbind(1,bcg$x)

# When, for example, using the Mandel-Paule estimate:
bcg_h <- pfunc(y=bcg_y, d=bcg_d, x=bcg_x)(dim(bcg_x)[1] -
  dim(bcg_x)[2])

set.seed(51351) # for reproducibility
random_g <- rnorm(dim(bcg_x)[2])
formulaR(y=bcg_y, d=bcg_d, h=bcg_h, g=random_g, x=bcg_x)

# The function can also be used when planing to perform
# a meta regression with no intercept, and only a singel
# covariate (i.e. dim(x) = 1).  In this case,
# the design matrix can simply be provided by a vector.
set.seed(51351) # for reproducibility
random_g <- rnorm(1)
formulaR(y=bcg_y, d=bcg_d, h=bcg_h, g=random_g, x=bcg$x)

# When performing a meta analysis, provide the function
# with a vector of 1s.
formulaR(y=bcg_y, d=bcg_d, h=bcg_h, g=random_g, x=rep(1,
  length(bcg_y)))
```

hConfidence *Inference: Based on methods of moments and maximum likelihood.*

Description

Calculates the so called Q-profiling confidence interval for the heterogeneity for data following a random effects meta regression model.

Usage

```
hConfidence(y, d, x, sgnf)
```

Arguments

y	k-vector of study responses.
d	k-vector of heteroscedasticity.
x	design k-p-matrix.
sgnf	significance levels.

Value

A data frame containing the bounds of the interval estimate.

Examples

```
bcg   <- bcgVaccineData()
bcg_y <- bcg$logrisk
bcg_d <- bcg$sdiv
bcg_s <- bcg$size
bcg_x <- cbind(1,bcg$x)
sgnf_lev <- c(0.01, 0.025, 0.05, 0.01)

set.seed(865287113) # for reproducibility

hConfidence(y=bcg_y, d=bcg_d, x=bcg_x, sgnf=0.025)
hConfidence(y=bcg_y, d=bcg_d, x=bcg_x, sgnf=sgnf_lev)
```

hEstimates *Point estimates: For the heterogeneity parameter*

Description

Returns a list of tau estimates based on different approximative methods. Different point estimates for the heterogeneity parameter are calculated: HD (Hedges), SL (DerSimonian-Laird), SJ (Sidik-Jonkman), MP (Mandel-Paule), ML (maximum likelihood), REML (restricted maximum-likelihood). Since any of these methods may fail to converge, there result may be 'NA' in this case.

Usage

```
hEstimates(y, d, x)
```

Arguments

y	study responses
d	heteroscedasticity
x	design matrix

Value

A data frame containing point estimates. Variables are 'type' and 'h'.

Examples

```
bcg   <- bcgVaccineData()
bcg_y <- bcg$logrisk
bcg_d <- bcg$sdiv
bcg_x <- cbind(1,bcg$x)
hEstimates(y=bcg_y, d=bcg_d, x=bcg_x)

# The implementation can also handle the case in which
# a meta regression is planed with no intercept and only a
# single covariate (i.e. dim(x) = 1).  In this case,
# the design matrix can simply be provided by a vector.
# (This makes no sense in this example and shall only prove
# feasibility)
hEstimates(y=bcg_y, d=bcg_d, x=bcg$x)

# When performing a meta analysis, provide the function
# with a vector of 1s.
hEstimates(y=bcg_y, d=bcg_d, x=rep(1, length(bcg_y)))
```

---

intervalEstimates *Interval estimates: For the regression coefficients*

Description

Interval estimates: For the regression coefficients

Usage

```
intervalEstimates(y, d, h_dat, x, sgnf)
```

Arguments

y	study responses.
d	heteroscedasticity.
h_dat	data frame of tau estimates.
x	design matrix.
sgnf	significance levels.

Examples

```
bcg   <- bcgVaccineData()
bcg_y <- bcg$logrisk
bcg_d <- bcg$sdiv
bcg_x <- cbind(1,bcg$x)
bcg_h <- hEstimates(y=bcg_y, d=bcg_d, x=bcg_x)
sgnf_lev <- c(0.01, 0.025, 0.05, 0.01)

intervalEstimates(y=bcg_y, d=bcg_d, h_dat=bcg_h, x=bcg_x, sgnf=0.025)
intervalEstimates(y=bcg_y, d=bcg_d, h_dat=bcg_h, x=bcg_x,
  sgnf=sgnf_lev)
```

---

joinPivotalCoefficients

Pivotal distributions: Extract pivots for regression coefficients

---

Description

Pivotal distributions: Extract pivots for regression coefficients

Usage

```
joinPivotalCoefficients(p0, p1)
```

Arguments

p0	pivotal stream without adjustment.
p1	pivatal stream with adjustment.

Examples

```
bcg   <- bcgVaccineData()
bcg_y <- bcg$logrisk
bcg_d <- bcg$sdiv
bcg_s <- bcg$size
bcg_x <- cbind(1,bcg$x)

set.seed(865287113)
pivUn <- pivotalStream(50, y=bcg_y, d=bcg_d, x=bcg_x,
```

```
  adjusted=FALSE)
set.seed(865287113)
  pivAd <- pivotalStream(50, y=bcg_y, d=bcg_d, x=bcg_x, s=bcg_s,
adjusted=TRUE)

pivr <- joinPivotalCoefficients(pivUn, pivAd)
```

---

joinPivotalHeterogeneity *Pivotal distributions: Extract pivots for heterogeneity*

Description

Pivotal distributions: Extract pivots for heterogeneity

Usage

```
joinPivotalHeterogeneity(p0 = NULL, p1 = NULL)
```

Arguments

p0	pivotal stream without adjustment.
p1	pivatal stream with adjustment.

Examples

```
bcg   <- bcgVaccineData()
bcg_y <- bcg$logrisk
bcg_d <- bcg$sdiv
bcg_s <- bcg$size
bcg_x <- cbind(1,bcg$x)

set.seed(865287113)
pivUn <- pivotalStream(50, y=bcg_y, d=bcg_d, x=bcg_x,
  adjusted=FALSE)
set.seed(865287113)
pivAd <- pivotalStream(50, y=bcg_y, d=bcg_d, x=bcg_x, s=bcg_s,
  adjusted=TRUE)

pivh <- joinPivotalHeterogeneity(pivUn, pivAd)
```

lenBoxByMethod *Plotting performance: Box plot of mean width*

Description

Plotting performance: Box plot of mean width

Usage

```
lenBoxByMethod(res)
```

Arguments

res	The collected results from a computer experiment.

Value

A plot object.

lenBoxByType *Plotting performance: Box plot of mean width*

Description

Plotting performance: Box plot of mean width

Usage

```
lenBoxByType(res)
```

Arguments

res	The collected results from a computer experiment.

Value

A plot object.

`lenDenByMethod` *Plotting performance: Density estimate of mean width*

Description

By method.

Usage

```
lenDenByMethod(res)
```

Arguments

`res`	The collected results from a computer experiment.

Value

A plot object.

`lenDenByType` *Plotting performance: Density estimate of mean width*

Description

By type.

Usage

```
lenDenByType(res)
```

Arguments

`res`	The collected results from a computer experiment.

Value

A plot object.

makeConfInt *Interval estimates: Generic function*

Description

Generic function to produce interval estimates of univariate parameters based on first order limit theory.

Usage

```
makeConfInt(sgn, pst, fct, crt, name)
```

Arguments

sgn	one significance level.
pst	point estimate.
fct	standard error.
crt	function for critical value computation.
name	string: name of the method.

Details

Function for symmetric confidence intervals based on standard deviations, point estimates, and quantile functions.

Can only handle a single significance level! See 'makeConfInts' for a more flexible solution.

makeConfInts *Interval estimates: Generic function*

Description

Generic function to produce interval estimates of univariate parameters based on first order limit theory.

Usage

```
makeConfInts(sgn, pst, fct, crt, name)
```

Arguments

sgn	one significance level.
pst	point estimate.
fct	standard error.
crt	function for critical value computation.
name	string: name of the method.

Details

Function for symmetric confidence intervals based on standard deviations, point estimates, and quantile functions.

`metagen` *Inference: Analysis of the data set*

Description

Runs all implemented methods and combines them in a neat summary.

Usage

```
metagen(y, d, x, sgnf, s = NULL, n,
  method = list("univariate", "multivariate"),
  adjusted = FALSE)
```

Arguments

`y`	k-vector of responses.
`d`	k-vector of heteroscedasticities.
`x`	design k-p-matrix.
`sgnf`	vector of significance levels.
`s`	k-vector of study responses. Default is NULL. If 'adjusted=TRUE', this value needs to be given.
`n`	draws from the pivotal distribution.
`method`	Default is 'list("univariate", "multivariate")'.
`adjusted`	: TRUE or FALSE

Value

The same return type as the skeleton 'metagenEmpty()'.

Examples

```
bcg   <- bcgVaccineData()
bcg_y <- bcg$logrisk
bcg_d <- bcg$sdiv
bcg_x <- cbind(1,bcg$x)
sgnf_lev <- c(0.01, 0.025, 0.05, 0.01)

set.seed(865287113) # for reproducibility

# Runs a standard analysis, use n=1000 in an actual
# analysis instead!
m1 <- metagen(y=bcg_y, d=bcg_d, x=bcg_x, sgnf=0.025, n=50)
```

```
m2 <- metagen(y=bcg_y, d=bcg_d, x=bcg_x, sgnf=sgnf_lev, n=50)

# Runs the methods based on generalised principles via an
# adjustment for the unknown heteroscedasticity.  Use
# n=1000 in an actual analysis instead!!
bcg_s <- bcg$size
m3 <- metagen(y=bcg_y, d=bcg_d, x=bcg_x, sgnf=0.025, s=bcg_s, n=50,
  adj=TRUE)
m4 <- metagen(y=bcg_y, d=bcg_d, x=bcg_x, sgnf=sgnf_lev, s=bcg_s,
  n=50, adj=TRUE)

if (!all(names(m1) == names(metagenEmpty()))) stop("Name clash")
if (!all(names(m2) == names(metagenEmpty()))) stop("Name clash")
if (!all(names(m3) == names(metagenEmpty()))) stop("Name clash")
if (!all(names(m4) == names(metagenEmpty()))) stop("Name clash")
```

`metagenEmpty` *Inference: Empty skeleton*

Description

Returns an empty skeleton that has the same return type as any other 'metagenSOMETHING' function.

Usage

```
metagenEmpty()
```

Examples

```
metagenEmpty()
```

`metagenGeneralised` *Inference: Based on generalised inference principles.*

Description

Inference: Based on generalised inference principles.

Usage

```
metagenGeneralised(y, d, x, sgnf, s = NULL, n,
  method = list("univariate", "multivariate"),
  adjusted = FALSE)
```

Arguments

y	k-vector of responses.
d	k-vector of heteroscedasticities.
x	design k-p-matrix.
sgnf	vector of significance levels
s	k-vector of study responses. No need to provide this, when 'adjusted==FALSE'. Default is NULL.
n	draws from the pivotal distribution.
method	Default is 'list("univariate", "multivariate")'.
adjusted	TRUE or FALSE. Default is FALSE.

Examples

```
bcg   <- bcgVaccineData()
bcg_y <- bcg$logrisk
bcg_d <- bcg$sdiv
bcg_x <- cbind(1,bcg$x)
sgnf_lev <- c(0.01, 0.025, 0.05, 0.01)

set.seed(865287113) # for reproducibility

# Runs a standard analysis, use n=1000 in an actual
# analysis instead!!
g1 <- metagenGeneralised(y=bcg_y, d=bcg_d, x=bcg_x, sgnf=0.025, n=50)
g2 <- metagenGeneralised(y=bcg_y, d=bcg_d, x=bcg_x, sgnf=sgnf_lev,
  n=50)

# Runs the methods based on generalised principles via an
# adjustment for the unknown heteroscedasticity.  Use n=1000 in an
# actual analysis instead!!
bcg_s <- bcg$size
g3 <- metagenGeneralised(y=bcg_y, d=bcg_d, x=bcg_x, sgnf=0.025,
  s=bcg_s, n=50, adj=TRUE)
g4 <- metagenGeneralised(y=bcg_y, d=bcg_d, x=bcg_x, sgnf=sgnf_lev,
  s=bcg_s, n=50, adj=TRUE)

# The implementation can also handle the case in which
# a meta regression is planed with no intercept and only a
# single covariate (i.e. dim(x) = 1).  In this case,
# the design matrix can simply be provided by a vector.
# (This makes no sense in this example and shall only proves
# feasibility)
g5 <- metagenGeneralised(y=bcg_y, d=bcg_d, x=bcg$x, sgnf=0.025, n=50)

# When performing a meta analysis, provide the function
# with a vector of 1s.
g6 <- metagenGeneralised(y=bcg_y, d=bcg_d, x=rep(1,length(bcg_y)),
  sgnf=0.025, n=50)
```

```
if (!all(names(g1) == names(metagenEmpty()))) stop("Name clash")
if (!all(names(g2) == names(metagenEmpty()))) stop("Name clash")
if (!all(names(g3) == names(metagenEmpty()))) stop("Name clash")
if (!all(names(g4) == names(metagenEmpty()))) stop("Name clash")
if (!all(names(g5) == names(metagenEmpty()))) stop("Name clash")
if (!all(names(g6) == names(metagenEmpty()))) stop("Name clash")
```

`metareg` *Inference: Based on methods of moments and maximum likelihood.*

Description

Calculates common statistics for point and confidence interval estimates for the heterogeneity and the regression coefficients of the random effects meta regression model based on the given data.

Usage

```
metareg(y, d, x, sgnf)
```

Arguments

`y`	k-vector of study responses.
`d`	k-vector of heteroscedasticity.
`x`	design k-p-matrix.
`sgnf`	significance levels.

Value

The same return type as the skeleton 'metagenEmpty()'.

Examples

```
bcg   <- bcgVaccineData()
bcg_y <- bcg$logrisk
bcg_d <- bcg$sdiv
bcg_s <- bcg$size
bcg_x <- cbind(1,bcg$x)
sgnf_lev <- c(0.01, 0.025, 0.05, 0.01)

set.seed(865287113) # for reproducibility

c1 <- metareg(y=bcg_y, d=bcg_d, x=bcg_x, sgnf=0.025)
c2 <- metareg(y=bcg_y, d=bcg_d, x=bcg_x, sgnf=sgnf_lev)

# When performing a meta analysis, provide the function
# with a vector of 1s.
if (!all(names(c1) == names(metagenEmpty()))) stop("Name clash")
if (!all(names(c2) == names(metagenEmpty()))) stop("Name clash")
```

`performance` *Running a computer experiment*

Description

Adding performance measures to the results

Usage

```
performance(results, b, h)
```

Arguments

`results`	Needs to be of the same type as, for example, the return value of the computer experiments 'experimentY', 'experimentD'.
`b`	true regression coefficients.
`h`	true heterogeneity.

Details

Calculating performance measurements from a computer experiment.

Value

Data frame containing performance measurements of inference methods based on the results of the computer experiment given by 'results'.

Examples

```
h_test <- 0.03
x_test <- cbind(1,1:7)
b_test <- c(.5, .25)
sgnf_test <- c(0.025, 0.01)

set.seed(5133568) # for reproducibility
d_test <- rchisq(7, df=0.02)

# In an actual computer experiment, use 'piv_draws=1000' instead!!
eY <- experimentY(n=5, h=h_test, d=d_test, x=x_test, b=b_test,
  sgnf=sgnf_test, piv_draws=50)

performance(results=eY, b=b_test, h=h_test)
```

performanceConfH *Running a computer experiment: Adding performance measures*

Description

Adding performance measurements to accumulated results of a computer experiment running multiple analysis of different simulated data following a random effects meta regression model.

Usage

```
performanceConfH(accum_int, true)
```

Arguments

accum_int	accumulated interval estimates. At least the following columns need to be present: lower and upper.
true	true parameter.

Details

Adds performance measurements to interval estimates of the heterogeneity.

Examples

```
# For an example, see the 'performance' function.
```

performanceConfR *Running a computer experiment: Adding performance measures*

Description

Adding performance measurements to accumulated results of a computer experiment running multiple analysis of different simulated data following a random effects meta regression model.

Usage

```
performanceConfR(accum_int, true)
```

Arguments

accum_int	accumulated interval estimates. At least the following columns need to be present: lower and upper and parameter.
true	true parameter.

Details

Adds performance measurements to interval estimates of the regression coefficients.

Examples

```
# For an example, see the 'performance' function.
```

`performancePointH` *Running a computer experiment: Adding performance measures*

Description

Adding performance measurements to accumulated results of a computer experiment running multiple analysis of different simulated data following a random effects meta regression model.

Usage

```
performancePointH(point, h)
```

Arguments

`point`	accumulated point estimates.
`h`	true parameter.

Details

Adds performance measurements to point estimates of the heterogeneity.

Examples

```
# For an example, see the 'performance' function.
```

`performancePointR` *Running a computer experiment: Adding performance measures*

Description

Adding performance measurements to accumulated results of a computer experiment running multiple analysis of different simulated data following a random effects meta regression model.

Usage

```
performancePointR(point, b)
```

Arguments

point	accumulated point estimates.
b	true parameter.

Details

Adds performance measurements to point estimates of the regression coefficients.

Examples

```
# For an example, see the 'performance' function.
```

---

pfunc *The p_delta(eta) function.*

Description

Returns the p-function.

Usage

```
pfunc(y, d, x)
```

Arguments

y	study responses.
d	heteroscedasticity.
x	design matrix.

Value

A vector valued function.

Examples

```
bcg   <- bcgVaccineData()
bcg_y <- bcg$logrisk
bcg_d <- bcg$sdiv
bcg_x <- cbind(1,bcg$x)
pfunc(y=bcg_y, d=bcg_d, x=bcg_x)

# Calculating the Mandel-Paule estimate:
pfunc(y=bcg_y, d=bcg_d, x=bcg_x)(dim(bcg_x)[1] - dim(bcg_x)[2])
```

`pivotalStream` *Steams of pivotal quantities of the regression coefficient*

Description

Algorithm for generating a steam of generalised pivotal quantities for the regression coefficients. If adjusted=FALSE, then no adjustments are made for the uncertainty in the heteroscedasticity estimates d. If adjusted=TRUE, then adjustments are performed. In this case, 's' needs to be provided.

Usage

```
pivotalStream(n, y, d, x, s = NULL,
  method = list("univariate", "multivariate"), adjusted)
```

Arguments

`n`	length of stream.
`y`	k-vector of responses.
`d`	k-vector of heteroscedasticity.
`x`	design (k,p)-matrix.
`s`	k-vector of study responses. No need to provide this, when adjusted=FALSE. Default is NULL.
`method`	A list. Used to choose the methods for calculating the pivotal quantities of the regression coefficients. Default is 'method=list("univariate", "multivariate")'.
`adjusted`	TRUE or FALSE. Default is FALSE.

Value

If method=="univariate" or method=="multivariate", then the return is a (p+1)-n-matrix. The first row contains pivotal quantities of the heterogeneity, the rest of the rows pivotal quantities of the regression coefficients. Each column is an independent draw.

If 'method==list("univariate", "multivariate")', then the return is a (2p+1)-n-matrix. Of each column, the first element is a pivotal for the heterogeneity, the next 'p' elements is a pivotal vector for the regression coefficients based on "univariate", the last 'p' elements are a pivotal vector for the regression coefficients based on "multivariate"

plotCoefficientInterval *Plot pivots: Interval estimates of the heterogeneity*

Description

Plot pivots: Interval estimates of the heterogeneity

Usage

```
plotCoefficientInterval(cnfr)
```

Arguments

cnfr	interval estimates of the heterogeneity.

plotDensityH *Pivotal distributions: Plot pivotal distribution of heterogeneity*

Description

Pivotal distributions: Plot pivotal distribution of heterogeneity

Usage

```
plotDensityH(pivh)
```

Arguments

pivh	pivotal stream with or without adjustment of independent draws of a pivotal quantity of the heterogeneity.

Examples

```
bcg   <- bcgVaccineData()
bcg_y <- bcg$logrisk
bcg_d <- bcg$sdiv
bcg_s <- bcg$size
bcg_x <- cbind(1,bcg$x)

set.seed(865287113)
pivUn <- pivotalStream(50, y=bcg_y, d=bcg_d, x=bcg_x,
  adjusted=FALSE)
set.seed(865287113)
pivAd <- pivotalStream(50, y=bcg_y, d=bcg_d, x=bcg_x, s=bcg_s,
  adjusted=TRUE)

pivh <- joinPivotalHeterogeneity(pivUn, pivAd)
plotDensityH(pivh)
```

plotDensityH2 *Pivotal distributions: Plot pivot density of the heterogeneity*

Description

Pivotal distributions: Plot pivot density of the heterogeneity

Usage

```
plotDensityH2(pivh)
```

Arguments

pivh	pivotal stream with or without adjustment of independent draws of a pivotal quantity of the heterogeneity.

Examples

```
bcg   <- bcgVaccineData()
bcg_y <- bcg$logrisk
bcg_d <- bcg$sdiv
bcg_x <- cbind(1,bcg$x)

set.seed(865287113)
pivUn <- pivotalStream(50, y=bcg_y, d=bcg_d, x=bcg_x,
  adjusted=FALSE)
pivh  <- joinPivotalHeterogeneity(pivUn)
plotDensityH2(pivh)
```

plotDensityIntercept *Pivotal distributions: Plot pivotal distribution of regression coefficients*

Description

Pivotal distributions: Plot pivotal distribution of regression coefficients

Usage

```
plotDensityIntercept(pivr)
```

Arguments

pivr	data frame of independent draws from of pivots.

Examples

```
bcg   <- bcgVaccineData()
bcg_y <- bcg$logrisk
bcg_d <- bcg$sdiv
bcg_s <- bcg$size
bcg_x <- cbind(1,bcg$x)

set.seed(865287113)
pivUn <- pivotalStream(50, y=bcg_y, d=bcg_d, x=bcg_x,
  adjusted=FALSE)
set.seed(865287113)
pivAd <- pivotalStream(50, y=bcg_y, d=bcg_d, x=bcg_x, s=bcg_s,
  adjusted=TRUE)

pivr <- joinPivotalCoefficients(pivUn, pivAd)
plotDensityIntercept(pivr)
```

---

plotDensityIntercept2 *Pivotal distributions: Plot pivotal distribution of regression coefficients*

Description

Pivotal distributions: Plot pivotal distribution of regression coefficients

Usage

```
plotDensityIntercept2(pivr)
```

Arguments

pivr	data frame of independent draws from of pivots.

Examples

```
bcg   <- bcgVaccineData()
bcg_y <- bcg$logrisk
bcg_d <- bcg$sdiv
bcg_s <- bcg$size
bcg_x <- cbind(1,bcg$x)

set.seed(865287113)
pivUn <- pivotalStream(50, y=bcg_y, d=bcg_d, x=bcg_x,
  adjusted=FALSE)
set.seed(865287113)
pivAd <- pivotalStream(50, y=bcg_y, d=bcg_d, x=bcg_x, s=bcg_s,
  adjusted=TRUE)

pivr <- joinPivotalCoefficients(pivUn, pivAd)
plotDensityIntercept2(pivr)
```

`plotDensitySlope` *Pivotal distributions: Plot pivotal distribution of regression coefficients*

Description

Pivotal distributions: Plot pivotal distribution of regression coefficients

Usage

```
plotDensitySlope(pivr)
```

Arguments

`pivr`	data frame of independent draws from of pivots.

Examples

```
bcg   <- bcgVaccineData()
bcg_y <- bcg$logrisk
bcg_d <- bcg$sdiv
bcg_s <- bcg$size
bcg_x <- cbind(1,bcg$x)

set.seed(865287113)
pivUn <- pivotalStream(50, y=bcg_y, d=bcg_d, x=bcg_x,
  adjusted=FALSE)
set.seed(865287113)
pivAd <- pivotalStream(50, y=bcg_y, d=bcg_d, x=bcg_x, s=bcg_s,
  adjusted=TRUE)

pivr <- joinPivotalCoefficients(pivUn, pivAd)
plotDensitySlope(pivr)
```

`plotDensitySlope2` *Pivotal distributions: Plot pivotal distribution of regression coefficients*

Description

Pivotal distributions: Plot pivotal distribution of regression coefficients

Usage

```
plotDensitySlope2(pivr)
```

Arguments

pivr	data frame of independent draws from of pivots.

Examples

```
bcg   <- bcgVaccineData()
bcg_y <- bcg$logrisk
bcg_d <- bcg$sdiv
bcg_s <- bcg$size
bcg_x <- cbind(1,bcg$x)

set.seed(865287113)
pivUn <- pivotalStream(50, y=bcg_y, d=bcg_d, x=bcg_x,
  adjusted=FALSE)
set.seed(865287113)
pivAd <- pivotalStream(50, y=bcg_y, d=bcg_d, x=bcg_x, s=bcg_s,
  adjusted=TRUE)

pivr <- joinPivotalCoefficients(pivUn, pivAd)
plotDensitySlope2(pivr)
```

---

plotHeterogeneityInterval *Plot pivots: Interval estimates of the heterogeneity*

Description

Plot pivots: Interval estimates of the heterogeneity

Usage

```
plotHeterogeneityInterval(cnfh)
```

Arguments

cnfh	interval estimates of the heterogeneity.

---

plotIntervalEstimates *Example: Plotting interval estimates*

Description

Plots a graphical representation of interval estimates in the data frame 'cnf' by type of method used for the estimation.

Usage

```
plotIntervalEstimates(cnf)
```

Arguments

cnf	data frame of interval estimates

Value

An object created by ggplot2.

---

plotStudyForest *Example: Plotting a forest plot of a data frame*

Description

Example: Plotting a forest plot of a data frame

Usage

```
plotStudyForest(dat)
```

Arguments

dat	data frame of study responses of binomial type.

Value

An object created by ggplot2.

Examples

```
bcg <- bcgVaccineData()
plotStudyForest(bcg)
```

plotStudyQfuncPfunc *Example: Plotting the q- and p-function from the dissertation*

Description

Example: Plotting the q- and p-function from the dissertation

Usage

```
plotStudyQfuncPfunc(y, d, x, n)
```

Arguments

y	a vector of responses.
d	a vector of heteroscedasticity.
x	a design matrix.
n	number of points to interpolate along.

Value

A list of objects created by ggplot2.

Examples

```
bcg   <- bcgVaccineData()
bcg_y <- bcg$logrisk
bcg_d <- bcg$sdiv
bcg_s <- bcg$size
bcg_x <- cbind(1,bcg$x)
p <- plotStudyQfuncPfunc(y=bcg_y, d=bcg_d, x=bcg_x, n=500)
p[1] # plot of the q-function
p[2] # plot of the p-funciton
```

plotStudySizes *Example: Plotting study sizes*

Description

Example: Plotting study sizes

Usage

```
plotStudySizes(dat)
```

Arguments

dat	data frame of study responses of binomial type.

Value

An object created by ggplot2.

Examples

```
bcg <- bcgVaccineData()
plotStudySizes(bcg)
```

---

plotStudyUnbalance *Example: Plotting study unbalances in group assignments*

Description

Example: Plotting study unbalances in group assignments

Usage

```
plotStudyUnbalance(dat)
```

Arguments

dat	data frame of study responses of binomial type.

Value

An object created by ggplot2.

---

qfunc *The q_delta(tau) function.*

Description

Returns the q-function.

Usage

```
qfunc(y, d, x)
```

Arguments

y	study responses.
d	heteroscedasticity.
x	design matrix.

Value

A vector valued function.

Examples

```
bcg   <- bcgVaccineData()
bcg_y <- bcg$logrisk
bcg_d <- bcg$sdiv
bcg_x <- cbind(1,bcg$x)
qfunc(y=bcg_y, d=bcg_d, x=bcg_x)
```

---

rB *Data generation: Log-risk-ration of a binomial-Gaussian model*

Description

Random draws of log risk ratios from a hierarchical binomial Gaussian model.

Usage

```
rB(n, h, s, a, r, x, b)
```

Arguments

n	number of draws.
h	heterogeneity.
s	study sizes.
a	balance of group assignments.
r	fixed risk in the treatment group.
x	design matrix.
b	regression coefficients.

Details

It is always assumed that at least one response in a study has happend, i.e., a response of 0 in a treatment or control group is rounded up to 1. Note that this may lead to an overestimation of small risks. If possible, make sure your sample sizes are large enough to compensate for this effect.

Value

A (2k,n) matrix. Each column is an independent draw.

Examples

```
h_test <- .03
x_test <- cbind(1,1:13)
b_test <- c(0.02, 0.03)
s_test <- rep(2000, 13)
a_test <- rep(.3, 13)
rB(n=10, h=h_test, s=s_test, a=a_test, r=.3, x=x_test, b=b_test)
```

`rBinomGauss` *Data generation: Sampling data of clinical trials*

Description

A random draw of a hierarchical binomial Gaussian model.

Usage

```
rBinomGauss(h, s, a, r, x, b)
```

Arguments

`h`	heterogeneity.
`s`	study sizes.
`a`	balance of group assignments.
`r`	fixed risk in the control group.
`x`	design matrix.
`b`	regression coefficients.

Details

It is always assumed that at least one response in a study has happend, i.e., a response of 0 in a treatment or control group is rounded up to 1. Note that this may lead to an overestimation of small risks. If possible, make sure your sample sizes are large enough to compensate for this effect.

You may work around this by increasing study sizes.

Value

A list containing the risk and a data frame with the studies.

Examples

```
h_test <- .03
s_test <- rep(2000, 13)
a_test <- rep(.3, 13)
x_test <- cbind(1,1:13)
b_test <- c(0.02, 0.03)
dat <- rBinomGauss(h=h_test, s=s_test, a=a_test, r=0.03 , x=x_test,
b=b_test)$study

if(!all(dim(dat) == c(dim(x_test)[1], 4))) stop("Wrong dimension")
```

---

rD *Data generation: Gaussian-Gaussian model*

Description

Random draws of heteroscedasticity responses of studies, where each study in a random effects meta regression model follows a Gaussian response. Thus D = (d * X) / (s-1) where X is chi-squared distributed.

Usage

```
rD(n, d, s)
```

Arguments

n	number of draws.
d	heteroscedasticity.
s	study sizes.

Value

A (k,n)-matrix. Each column is an independent draw.

Examples

```
d_test = rchisq(13, df=0.02)
s_test = rep(100, 13)
rD(n=10, d=d_test, s=s_test)
```

regressionEstimates *Point estimates: For the regression coefficients*

Description

Calculates point estimates for the regression coefficient for given point estimates of the variance components 'd' and a data frame of different estimates of the heterogeneity 'h'.

Usage

```
regressionEstimates(y, d, h_dat, x)
```

Arguments

y	study responses, k-vector of responses.
d	heteroscedasticity, k-vector of heteroscedasticities.
h_dat	Here, 'h_dat' should be a data frame with variables 'type' and 'h'. Thus, one may use h_dat = hEstimates(y, d, x).
x	design matrix, k-p-matrix.

Value

A list of estimates for the regression coefficients.

Here, 'h_dat' should be a data frame with variables 'type' and 'h', thus, we may use h_dat = hEstimates(y, d, x)

Examples

```
bcg   <- bcgVaccineData()
bcg_y <- bcg$logrisk
bcg_d <- bcg$sdiv
bcg_x <- cbind(1,bcg$x)
bcg_h <- hEstimates(y=bcg_y, d=bcg_d, x=bcg_x)
regressionEstimates(y=bcg_y, d=bcg_d, h_dat=bcg_h, x=bcg_x)
```

render *Render plot: To PDF*

Description

Renders obj into a pdf-file of name: path++name. Neat feature is that the default size in A4. Simply use the 'scale' parameter to adjust the size of the plot to a fraction of a page.

Usage

```
render(name, plotObj, path, scale = 1, height = 11.6,
  width = 8.2)
```

Arguments

name	Should be self explanatory.
plotObj	Should be self explanatory.
path	Should be self explanatory.
scale	Should be self explanatory.
height	Should be self explanatory.
width	Should be self explanatory.

---

renderSVG *Render plot: To SVG*

Description

Renders obj into a svg-file of name: path++name. Neat feature is that the default size in A4. Simply use the ‘scale‘ parameter to adjust the size of the plot to a fraction of a page.

Usage

```
renderSVG(name, plotObj, path, scale = 1, height = 11.6,
  width = 8.2)
```

Arguments

name	Should be self explanatory.
plotObj	Should be self explanatory.
path	Should be self explanatory.
scale	Should be self explanatory.
height	Should be self explanatory.
width	Should be self explanatory.

rY *Data generation: Gaussian-Gaussian model*

Description

Random draws of response vectors y following the distribution of a random effects meta regression model. Each column is an independent draw.

Usage

```
rY(n, h, d, x, b)
```

Arguments

n	number of draws.
h	heterogeneity.
d	heteroscedasticity.
x	design matrix.
b	regression coefficients.

Value

A (k,n)-matrix. Each column is an independent draw.

Examples

```
x_test = cbind(1,1:13)
h_test = .03
d_test = rchisq(13, df=0.02)
b_test = c(0.02, 0.03)
rY(n=10, h=h_test, d=d_test, x=x_test, b=b_test)
```

sctBias *Plotting performance: Scatter plots against heterogeneity*

Description

Scatter plots of heterogeneity and bias.

Usage

```
sctBias(res, ...)
```

Arguments

res	The collected results from a computer experiment.
...	further arguments to scale_y_continuous

Value

A plot object.

---

sctMSE *Plotting performance: Scatter plots against heterogeneity*

Description

Scatter plots of heterogeneity and mean squared error.

Usage

```
sctMSE(res, ...)
```

Arguments

res	The collected results from a computer experiment.
...	further arguments to scale_y_continuous

Value

A plot object.

---

sctSD *Plotting performance: Scatter plots against heterogeneity*

Description

Scatter plots of heterogeneity and standard deviation.

Usage

```
sctSD(res, ...)
```

Arguments

res	The collected results from a computer experiment.
...	further arguments to scale_y_continuous

Value

A plot object.

sctVersusC *Plotting performance: Scatter plot against heterogeneity*

Description

Plotting performance: Scatter plot against heterogeneity

Usage

```
sctVersusC(res)
```

Arguments

res	The collected results from a computer experiment.

Value

A plot object.

sctVersusH *Plotting performance: Scatter plot against heterogeneity*

Description

Plotting performance: Scatter plot against heterogeneity

Usage

```
sctVersusH(res)
```

Arguments

res	The collected interval results from a computer experiment.

Value

A plot object.

sdmByMethod *Plotting performance: Scatter plot against heterogeneity*

Description

Plotting performance: Scatter plot against heterogeneity

Usage

```
sdmByMethod(res)
```

Arguments

res	The collected results from a computer experiment.

Value

A plot object.

sdmByType *Plotting performance: Scatter plot against heterogeneity*

Description

Plotting performance: Scatter plot against heterogeneity

Usage

```
sdmByType(res)
```

Arguments

res	The collected results from a computer experiment.

Value

A plot object.

sdsByMethod *Plotting performance: Scatter plot against heteroscedasticity*

Description

Plotting performance: Scatter plot against heteroscedasticity

Usage

```
sdsByMethod(res)
```

Arguments

res	The collected results from a computer experiment.

Value

A plot object.

sdsByType *Plotting performance: Scatter plot against heteroscedasticity*

Description

Plotting performance: Scatter plot against heteroscedasticity

Usage

```
sdsByType(res)
```

Arguments

res	The collected results from a computer experiment.

Value

A plot object.

`setupExperiment` *Running a computer experiment in batch mode*

Description

Sets up a computer experiment evaluating the performance of different inference methods in the random effects meta regression model.

Usage

```
setupExperiment(name, seed, n, resolution, bounds, x, b,
  sgnf, piv_draws, ...)
```

Arguments

`name`	Reference name for the experiment.
`seed`	Random seed for the experiment.
`n`	number of simulations to at each parameter configuration.
`resolution`	list of number of parameter configurations in each design, e.g. resolution=list(h=5L, d=3L)
`bounds`	list of parameter bounds used for experimental design, e.g. bounds=list(h=c(0,1), d=c(0.001, 2), s=c(200L, 2000L)) where - h : bounds of the heterogeneity. - d : bounds of the heteroscedasticity. - a : bounds of the balancing factor of group assignments. - s : bounds of the study sizes. - r : fixed risk in the control.
`x`	design matrix.
`b`	regression coefficients.
`sgnf`	levels of significance.
`piv_draws`	number of pivotal draws.
`...`	further arguments to makeExperimentRegistry, e.g. file.dir=tempfile().

Value

The registry.

yvec *Data generation: Sampling data of clinical trials*

Description

Calculates log risk ratios from a study in the right format.

Usage

```
yvec(study)
```

Arguments

study	Study data of a clinical trial with binomial outcomes.

Examples

```
h_test <- .03
x_test <- cbind(1,1:13)
b_test <- c(0.02, 0.03)
s_test <- rep(2000, 13)
a_test <- rep(.3, 13)
rBinomGauss(  h=h_test, s=s_test, a=a_test, r=0.03
            , x=x_test, b=b_test)$study -> test
yvec(test)
dvec(test)
```

Index