Package: mcrPioda (via r-universe)

October 16, 2026

Version 1.3.4

Date 2024-08-27

Title Method Comparison Regression - Mcr Fork for M- And MM-Deming Regression

Maintainer Giorgio Pioda <gfwp@ticino.com>

Copyright This package includes code from the 'clinfun' library owned by Ventatraman E. Seshan (ktau.r and ktau.f).

Depends R (>= 3.0.0), parallel, robslopes

Imports stats, graphics, grDevices, methods, rrcov, mixtools

Description Regression methods to quantify the relation between two measurement methods are provided by this package. In particular it addresses regression problems with errors in both variables and without repeated measurements. It implements the Clinical Laboratory Standard International (CLSI) recommendations (see J. A. Budd et al. (2018, <https://clsi.org/standards/products/method-evaluation/documents/ep09/>) for analytical method comparison and bias estimation using patient samples. Furthermore, algorithms for Theil-Sen and equivariant Passing-Bablok estimators are implemented, see F. Dufey (2020, <doi:10.1515/ijb-2019-0157>) and J. Raymaekers and F. Dufey (2022, <arXiv:2202:08060>). Further the robust M-Deming and MM-Deming (experimental) are available, see G. Pioda (2021, <arXiv:2105:04628>). A comprehensive overview over the implemented methods and references can be found in the manual pages 'mcrPioda-package' and 'mcreg'.

License GPL (>= 3)

Collate ``mcrMisc.r'' ``mcLinReg.r'' ``mcDeming.r'' ``mcWDeming.r'' ``mcPaBaLarge.r'' ``mcPaBa.r'' ``mcPBequi.r'' ``mcCalcCI.r'' ``mcCalcTstar.r'' ``mcBootstrap.r'' ``MCResultMethods.r'' ``MCResult.r'' ``MCResultAnalyticalMethods.r'' ``MCResultAnalytical.r'' ``MCResultJackknifeMethods.r'' ``MCResultJackknife.r'' ``MCResultResamplingMethods.r'' ``MCResultResampling.r'' ``MCResultBCaMethods.r'' ``MCResultBCa.r''

`` `mcrInterface.r" `` `mcrCompareFit.r" `` `mcrIncludeLegend.r" `` `zzz.r" `` `ktau.r" `` `mcMDeming.r" `` `mcMMDeming.r" `` `mcNgMMDeming.r" `` `mcPiMMDeming.r"

RoxygenNote 7.3.2

LazyData true

NeedsCompilation yes

Author Giorgio Pioda [aut, cre] (<https://orcid.org/0000-0002-8714-8678>), Sergej Potapov [aut] (<https://orcid.org/0009-0002-6251-0279>), Fabian Model [aut], Andre Schuetzenmeister [aut] (<https://orcid.org/0000-0002-2964-5502>), Ekaterina Manuilova [aut], Florian Dufey [aut] (<https://orcid.org/0000-0001-6467-8556>), Jakob Raymaekers [aut] (<https://orcid.org/0000-0002-2093-3137>), Venkatraman E. Seshan [ctb], Roche [cph, fnd]

Repository https://cran.r-universe.dev

Date/Publication 2024-08-28 02:50:16 UTC

RemoteUrl https://github.com/cran/mcrPioda

RemoteRef HEAD

RemoteSha 409357adfc52ea19c15a79c9343f158fb1a658fc

Contents

mcrPioda-package 4
calcDiff 5
compareFit 5
creatinine 6
includeLegend 6
mc.analytical.ci 8
mc.bootstrap 9
mc.calc.bca 11
mc.calc.quant 12
mc.calc.quantile 12
mc.calc.Student 13
mc.calc.tboot 14
mc.calcAngleMat 15
mc.calcAngleMat.R 15
mc.calcLinnetCI 16
mc.calcTstar 16
mc.deming 17
mc.linreg 18
mc.make.CIframe 19
mc.mdemingConstCV 20
mc.mmdemingConstCV 21
mc.mmNgdemingConstCV 22

mc.mmPidemingConstCV . . . 23
mc.paba . . . 24
mc.paba.LargeData . . . 25
mc.PBequi . . . 27
mc.wdemingConstCV . . . 28
mc.wlinreg . . . 29
mcreg . . . 29
MCResult-class . . . 35
MCResult.calcBias . . . 36
MCResult.calcCUSUM . . . 37
MCResult.calcPaBaTiesRatio . . . 38
MCResult.calcResponse . . . 39
MCResult.getCoefficients . . . 40
MCResult.getData . . . 40
MCResult.getErrorRatio . . . 41
MCResult.getFitted . . . 41
MCResult.getRegmethod . . . 42
MCResult.getResiduals . . . 42
MCResult.getWeights . . . 43
MCResult.initialize . . . 43
MCResult.plot . . . 44
MCResult.plotBias . . . 47
MCResult.plotDifference . . . 50
MCResult.plotResiduals . . . 52
MCResult.printSummary . . . 54
MCResultAnalytical-class . . . 54
MCResultAnalytical.calcResponse . . . 55
MCResultAnalytical.initialize . . . 56
MCResultAnalytical.printSummary . . . 57
MCResultBCa-class . . . 57
MCResultBCa.bootstrapSummary . . . 59
MCResultBCa.calcResponse . . . 59
MCResultBCa.initialize . . . 60
MCResultBCa.plotBootstrapCoefficients . . . 61
MCResultBCa.plotBootstrapT . . . 62
MCResultBCa.plotBoxEllipses . . . 62
MCResultBCa.printSummary . . . 63
MCResultJackknife-class . . . 63
MCResultJackknife.calcResponse . . . 65
MCResultJackknife.getJackknifeIntercept . . . 65
MCResultJackknife.getJackknifeSlope . . . 66
MCResultJackknife.getJackknifeStatistics . . . 66
MCResultJackknife.getRJIF . . . 67
MCResultJackknife.initialize . . . 67
MCResultJackknife.plotwithRJIF . . . 68
MCResultJackknife.printSummary . . . 69
MCResultResampling-class . . . 70
MCResultResampling.bootstrapSummary . . . 71

MCResultResampling.calcResponse 72
MCResultResampling.initialize 72
MCResultResampling.plotBootstrapCoefficients 74
MCResultResampling.plotBootstrapT 75
MCResultResampling.plotBoxEllipses 75
MCResultResampling.printSummary 76
newMCResult 76
newMCResultAnalytical 77
newMCResultBCa 78
newMCResultJackknife 80
newMCResultResampling 81

Index **83**

`mcrPioda-package` *Method Comparison Regression - Mcr Fork for M- and MM-Deming Regression*

Description

Regression methods to quantify the relation between two measurement methods are provided by this package. In particular it addresses regression problems with errors in both variables and without repeated measurements. It implements the CLSI recommendations for analytical method comparison and bias estimation using patient samples.

The main function for performing regression analysis is `mcreg`. Various functions for summarizing and plotting regression results are provided (see examples in `mcreg`).

For user site testing (installation verification) please use the test case suite provided with the package. The test case suite can be run by sourcing the 'runalltests.R' script in the 'unitTests' folder. It requires the XML and Runit packages.

Details

Package:	mcrPioda
Type:	Package
Version:	1.3.3
Date:	2024-05-30
License:	GPL 3
LazyLoad:	yes

Author(s)

Giorgio Pioda <gfwp@ticino.com>, Sergej Potapov <sergej.potapov@roche.com>, Fabian Model <fabian.model@roche.com>, Andre Schuetzenmeister <andre.schuetzenmeister@roche.com>, Ekaterina Manuilova <ekaterina.manuilova@roche.com>, Florian Dufey <florian.dufey@roche.com>, Jakob Raymaekers <jakob.raymaekers@kuleuven.com>

References

CLSI EP09 https://clsi.org/

`calcDiff` *Calculate difference between two numeric vectors that gives exactly zero for very small relative differences.*

Description

Calculate difference between two numeric vectors that gives exactly zero for very small relative differences.

Usage

```
calcDiff(X, Y, EPS = 1e-12)
```

Arguments

`X`	first number
`Y`	second number
`EPS`	relative difference equivalent to zero

Value

difference

`compareFit` *Graphical Comparison of Regression Parameters and Associated Confidence Intervals*

Description

Graphical comparison of regression parameters (intercept and slope) and their associated 100(1-alpha)% confidence intervals for multiple fitted models of 'MCResult' sub-classes.

Usage

```
compareFit(...)
```

Arguments

`...`	list of fitted models, i.e. objects of "MCResult" sub-classes.

Value

No return value, instead a plot is generated

Examples

```
library("mcrPioda")
data("creatinine", package="mcrPioda")
fit.lr <- mcreg(as.matrix(creatinine), method.reg="LinReg", na.rm=TRUE)
fit.wlr <- mcreg(as.matrix(creatinine), method.reg="WLinReg", na.rm=TRUE)
compareFit( fit.lr, fit.wlr )
```

---

creatinine *Comparison of blood and serum creatinine measurement*

Description

This data set gives the blood and serum preoperative creatinine measurements in 110 heart surgery patients.

Usage

```
creatinine
```

Format

A data frame containing 110 observations with serum and plasma creatinine measurements in mg/dL for each sample.

---

includeLegend *Include Legend*

Description

Include legend in regression plot (function `plot()`) or in bias plot (function `plotBias ()`) with two or more lines.

Usage

```
includeLegend(
  models = list(),
  digits = 2,
  design = paste(1:2),
  place = c("topleft", "topright", "bottomleft", "bottomright"),
  colors,
  lty = rep(1, length(models)),
  lwd = rep(2, length(models)),
  box.lty = "blank",
  cex = 0.8,
  bg = "white",
```

```
  inset = c(0.01, 0.01),
  bias = FALSE,
  model.names = NULL,
  ...
)
```

Arguments

models	list of length n with Objects of class "MCResult".
digits	number of digits in Coefficients.
design	type of legend design. There are two possible designs: "1" and "2" (See example).
place	place for Legend: "topleft","topright","bottomleft" or "bottomright".
colors	vector of length n with color of regression lines.
lty	vector of length n with type of regression lines.
lwd	vector of length n with thickness of regression lines.
box.lty	box line-type
cex	numeric value representing the plotting symbol magnification factor
bg	the background-color of the legend box
inset	inset distance(s) from the margins as a fraction of the plot region when legend is placed by keyword.
bias	logical value. If bias = TRUE, it will be drawn a legend for `plotBias()` function.
model.names	legend names for different models. If NULL the regression type will be used.
...	other parameters of function legend().

Value

Legend in plot.

See Also

plot.mcr, plotBias, plotResiduals, plotDifference, compareFit

Examples

```
#library("mcrPioda")

 data(creatinine,package="mcrPioda")
 x <- creatinine$serum.crea
 y <- creatinine$plasma.crea

 m1 <- mcreg(x,y,method.reg="Deming", mref.name="serum.crea",
                                       mtest.name="plasma.crea", na.rm=TRUE)
 m2 <- mcreg(x,y,method.reg="WDeming", method.ci="jackknife",
                                        mref.name="serum.crea",
```

```
                                        mtest.name="plasma.crea", na.rm=TRUE)

plot(m1,  XLIM=c(0.5,3),YLIM=c(0.5,3), Legend=FALSE,
                        Title="Deming vs. weighted Deming regression",
                        Points.pch=19,ci.area=TRUE, ci.area.col=grey(0.9),
                        identity=FALSE, Grid=FALSE, Sub="")
plot(m2, ci.area=FALSE, ci.border=TRUE, ci.border.col="red3",
                        reg.col="red3", Legend=FALSE,add=TRUE,
                        Points=FALSE, identity=FALSE, Grid=FALSE)

includeLegend(place="topleft",models=list(m1,m2),
                        colors=c("darkblue","red"), design="1", digits=2)
```

`mc.analytical.ci` *Analytical Confidence Interval*

Description

Calculate wald confidence intervals for intercept and slope given point estimates and standard errors.

Usage

```
mc.analytical.ci(b0, b1, se.b0, se.b1, n, alpha)
```

Arguments

`b0`	point estimate of intercept.
`b1`	point estimate of slope.
`se.b0`	standard error of intercept.
`se.b1`	standard error of slope.
`n`	number of observations.
`alpha`	numeric value specifying the 100(1-alpha)% confidence level for the confidence interval (Default is 0.05).

Value

2x4 matrix of estimates and confidence intervals for intercept and slope.

mc.bootstrap *Resampling estimation of regression parameters and standard errors.*

Description

Generate jackknife or (nested-) bootstrap replicates of a statistic applied to data. Only a nonparametric balanced design is possible. For each sample calculate point estimations and standard errors for regression coefficients.

Usage

```
mc.bootstrap(
 method.reg = c("LinReg", "WLinReg", "Deming", "WDeming", "PaBa", "PaBaLarge", "TS",
    "PBequi", "MDeming", "MMDeming", "NgMMDeming", "PiMMDeming"),
  jackknife = TRUE,
  bootstrap = c("none", "bootstrap", "nestedbootstrap"),
  X,
  Y,
  error.ratio,
  nsamples = 1000,
  priorSlope = 1,
  priorIntercept = 0,
  kM = 1.345,
  tauMM = 4.685,
  bdPoint = 0.5,
  nnested = 25,
  iter.max = 30,
  threshold = 1e-08,
  NBins = 1e+06,
  slope.measure = c("radian", "tangent")
)
```

Arguments

method.reg	Regression method. It is possible to choose between five regression types: "LinReg" - ordinary least square regression, "WLinReg" - weighted ordinary least square regression,"Deming" - Deming regression, "WDeming" - weighted Deming regression, "PaBa" - Passing-Bablok regression. "WDeming" - weighted Deming regression, "MDeming" - weighted M-Deming regression, "MMDeming" - weighted MM-Deming regression,"MMDeming" - Passing-Bablok regression. "NgMMDeming" - new generation MM-Deming regression,"NgMMDeming" - Passing-Bablok regression. "PiMMDeming" - prior informed MM-Deming regression,"PiMMDeming" - Passing-Bablok regression.
jackknife	Logical value. If TRUE - Jackknife based confidence interval estimation method.
bootstrap	Bootstrap based confidence interval estimation method.
X	Measurement values of reference method

Y	Measurement values of test method
error.ratio	Ratio between squared measurement errors of reference- and test method, necessary for Deming regression. Default 1.
nsamples	Number of bootstrap samples.
priorSlope	starting slope value for PiMMDeming, default priorSlope = 1
priorIntercept	starting intercept value for PiMMDeming, default priorIntercept = 0
kM	Huber's k for the M weighting, default kM = 1.345
tauMM	Tukey's tau for bisquare redescending weighting function, default tauMM = 4,685
bdPoint	Proportion of data points selected for the highly robust M regression used for the determination of the starting parameters. Default 0.5.
nnested	Number of nested bootstrap samples.
iter.max	maximum number of iterations for weighted Deming iterative algorithm.
threshold	Numerical tolerance for weighted Deming iterative algorithm convergence.
NBins	number of bins used when 'reg.method="PaBaLarge"' to classify each slope in one of 'NBins' bins of constant slope angle covering the range of all slopes.
slope.measure	angular measure of pairwise slopes used for exact PaBa regression (see mcreg for details). "radian" - for data sets with even sample numbers median slope is calculated as average of two central slope angles. "tangent" - for data sets with even sample numbers median slope is calculated as average of two central slopes (tan(angle)).

Value

a list consisting of

glob.coef	Numeric vector of length two with global point estimations of intercept and slope.
glob.sigma	Numeric vector of length two with global estimations of standard errors of intercept and slope.
xmean	Global (weighted-)average of reference method values.
B0jack	Numeric vector with point estimations of intercept for jackknife samples. The i-th element contains point estimation for data set without i-th observation
B1jack	Numeric vector with point estimations of slope for jackknife samples. The i-th element contains point estimation for data set without i-th observation
B0	Numeric vector with point estimations of intercept for each bootstrap sample. The i-th element contains point estimation for i-th bootstrap sample.
B1	Numeric vector with point estimations of slope for each bootstrap sample. The i-th element contains point estimation for i-th bootstrap sample.
MX	Numeric vector with point estimations of (weighted-)average of reference method values for each bootstrap sample. The i-th element contains point estimation for i-th bootstrap sample.

sigmaB0	Numeric vector with estimation of standard error of intercept for each bootstrap sample. The i-th element contains point estimation for i-th bootstrap sample.
sigmaB1	Numeric vector with estimation of standard error of slope for each bootstrap sample. The i-th element contains point estimation for i-th bootstrap sample.
nsamples	Number of bootstrap samples.
nnested	Number of nested bootstrap samples.
cimeth	Method of confidence interval calculation (bootstrap).
npoints	Number of observations.

Author(s)

Ekaterina Manuilova <ekaterina.manuilova@roche.com>, Fabian Model <fabian.model@roche.com>, Sergej Potapov <sergej.potapov@roche.com>

References

Efron, B., Tibshirani, R.J. (1993) *An Introduction to the Bootstrap*. Chapman and Hall. Carpenter, J., Bithell, J. (2000) Bootstrap confidence intervals: when, which, what? A practical guide for medical statisticians. *Stat Med*, **19 (9)**, 1141–1164.

---

mc.calc.bca *Bias Corrected and Accelerated Resampling Confidence Interval*

Description

Calculate resampling BCa confidence intervals for intercept, slope or bias given a vector of bootstrap and jackknife point estimates.

Usage

```
mc.calc.bca(Xboot, Xjack, xhat, alpha)
```

Arguments

Xboot	vector of point estimates for bootstrap samples. The i-th element contains point estimate of the i-th bootstrap sample.
Xjack	vector of point estimates for jackknife samples. The i-th element contains point estimate of the dataset without i-th observation.
xhat	point estimate for the complete data set (scalar).
alpha	numeric value specifying the 100(1-alpha)% confidence level for the confidence interval (Default is 0.05).

Value

a list with elements

`est`	point estimate for the complete data set (xhat).
`CI`	confidence interval for point estimate.

References

Carpenter, J., Bithell, J. (2000) Bootstrap confidence intervals: when, which, what? A practical guide for medical statisticians. *Stat Med*, **19 (9)**, 1141–1164.

`mc.calc.quant` *Quantile Calculation for BCa*

Description

We are using the R default (SAS (type=3) seems bugged) quantile calculation instead of the quantile function described in Effron&Tibshirani.

Usage

```
mc.calc.quant(X, alpha)
```

Arguments

`X`	numeric vector.
`alpha`	probability

Value

alpha-quantile of vector X.

`mc.calc.quantile` *Quantile Method for Calculation of Resampling Confidence Intervals*

Description

Calculate bootstrap confidence intervals for intercept, slope or bias given the vector of bootstrap point estimates.

Usage

```
mc.calc.quantile(Xboot, alpha)
```

Arguments

Xboot	vector of point estimates for bootstrap samples. The i-th element contains point estimate of the i-th bootstrap sample.
alpha	numeric value specifying the 100(1-alpha)% confidence level for the confidence interval (Default is 0.05).

Value

a list with elements

est	median of bootstrap point estimates Xboot.
CI	confidence interval for point estimate 'est', calculated as quantiles.

References

B. Efron and RJ. Tibshirani (1994) An Introduction to the Bootstrap. *Chapman & Hall*.

---

mc.calc.Student *Student Method for Calculation of Resampling Confidence Intervals*

Description

Calculate bootstrap confidence intervals for intercept, slope or bias given a vector of bootstrap point estimates.

Usage

```
mc.calc.Student(Xboot, xhat, alpha, npoints)
```

Arguments

Xboot	vector of point estimates for each bootstrap sample. The i-th element contains the point estimate of the i-th bootstrap sample.
xhat	global point estimate for which the confidence interval shall be computed.
alpha	numeric value specifying the 100(1-alpha)% confidence level for the confidence interval (Default is 0.05).
npoints	number of points used for the regression analysis.

Value

a list with elements

est	the point estimate xhat
se	standard deviation computed from bootstrap point estimates Xboot
CI	Confidence interval for point estimate xhat, calculated as $xhat + / - qt(1 - alpha, n - 2) * sd$.

References

Carpenter, J., Bithell, J. (2000) Bootstrap confidence intervals: when, which, what? A practical guide for medical statisticians. *Stat Med*, **19 (9)**, 1141–1164.

`mc.calc.tboot` *Bootstrap-t Method for Calculation of Resampling Confidence Intervals*

Description

Calculate resampling confidence intervals for intercept, slope or bias with t-Boot method given a vector of bootstrap point estimates and a vector of bootstrap standard deviations.

Usage

```
mc.calc.tboot(Xboot, Sboot, xhat, shat, alpha)
```

Arguments

`Xboot`	vector of point estimates for bootstrap sample. The i-th element contains the point estimate for the i-th bootstrap sample.
`Sboot`	vector of standard deviations for each bootstrap sample. It should be estimated with any analytical method or nonparametric with nested bootstrap.
`xhat`	point estimate for the complete data set (scalar).
`shat`	estimate of standard deviation for the complete data set (scalar).
`alpha`	numeric value specifying the 100(1-alpha)% confidence level for the confidence interval (Default is 0.05).

Value

a list with elements

`est`	point estimate for the complete data set (xhat).
`se`	estimate of standard deviation for the complete data set (shat).
`CI`	confidence interval for the point estimate.

References

Carpenter, J., Bithell, J. (2000) Bootstrap confidence intervals: when, which, what? A practical guide for medical statisticians. *Stat Med*, **19 (9)**, 1141–1164.

mc.calcAngleMat *Calculate Matrix of All Pair-wise Slope Angles*

Description

This version is implemented in C for computational efficiency.

Usage

```
mc.calcAngleMat(X, Y, posCor = TRUE)
```

Arguments

X	measurement values of reference method.
Y	measurement values of test method.
posCor	should algorithm assume positive correlation, i.e. symmetry around slope 1?

Value

Upper triangular matrix of slopes for all point combinations. Slopes in radian.

mc.calcAngleMat.R *Calculate Matrix of All Pair-wise Slope Angles*

Description

This is a very slow R version. It should not be called except for debugging purposes.

Usage

```
mc.calcAngleMat.R(X, Y, posCor = TRUE)
```

Arguments

X	measurement values of reference method.
Y	measurement values of test method.
posCor	should the algorithm assume positive correlation, i.e. symmetry around slope 1?

Value

Upper triangular matrix of slopes for all point combinations. Slopes in radian.

`mc.calcLinnetCI` *Jackknife Confidence Interval*

Description

Calculate Jackknife confidence intervals for intercept, slope or bias given of vector of jackknife point estimates and global point estimate.

Usage

```
mc.calcLinnetCI(Xjack, xhat, alpha = 0.05)
```

Arguments

`Xjack`	vector of point estimates for jackknife samples. The i-th element contains point estimate for the dataset without the i-th observation.
`xhat`	point estimate for the complete data set (scalar).
`alpha`	numeric value specifying the 100(1-alpha)% confidence level for the confidence interval (Default is 0.05).

Value

a list with elements

`est`	point estimate for the complete data set (scalar).
`se`	standard deviation of point estimate calculated with Jackknife Method.
`CI`	confidence interval for point estimate.

References

Linnet, K. (1993) Evaluation of Regression Procedures for Methods Comparison Studies. *CLIN. CHEM.* **39/3**, 424–432.

`mc.calcTstar` *Compute Resampling T-statistic.*

Description

Compute Resampling T-statistic. for Calculation of t-Bootstrap Confidence Intervals.

Usage

```
mc.calcTstar(
  .Object,
  x.levels,
  iter.max = 30,
  threshold = 1e-06,
  kM = 1.345,
  tauMM = 4.685,
  priorSlope = 1,
  priorIntercept = 0,
  bdPoint = 0.5
)
```

Arguments

.Object	object of class "MCResultResampling".
x.levels	a numeric vector of clinical decision points of interest.
iter.max	maximal number of iterations for calculation of weighted Deming regression.
threshold	threshold for calculation of weighted Deming regression.
kM	Huber's k for the M weighting, default kM = 1.345
tauMM	Tukey's tau for bisquare redescending weighting function, default tauMM = 4,685
priorSlope	starting slope value for PiMMDeming, default priorSlope = 1
priorIntercept	starting intercept value for PiMMDeming, default priorIntercept = 0
bdPoint	Proportion of data points selected for the highly robust M regression used for the determination of the starting parameters. Default 0.5

Value

Tstar numeric vector containing resampling pivot statistic.

References

Carpenter J., Bithell J. Bootstrap confidence intervals: when, which, what? A practical guide for medical statisticians. Stat Med, 19 (9), 1141-1164 (2000).

---

mc.deming *Calculate Unweighted Deming Regression and Estimate Standard Errors*

Description

Calculate Unweighted Deming Regression and Estimate Standard Errors

Usage

```
mc.deming(X, Y, error.ratio)
```

Arguments

X	measurement values of reference method.
Y	measurement values of test method.
error.ratio	ratio of measurement error of reference method to measurement error of test method.

Value

a list with elements

b0	intercept.
b1	slope.
se.b0	respective standard error of intercept.
se.b1	respective standard error of slope.
xw	average of reference method values.

References

Linnet K. Evaluation of Regression Procedures for Methods Comparison Studies. CLIN. CHEM. 39/3, 424-432 (1993).

Linnet K. Estimation of the Linear Relationship between the Measurements of two Methods with Proportional Errors. STATISTICS IN MEDICINE, Vol. 9, 1463-1473 (1990).

---

mc.linreg *Calculate ordinary linear Regression and Estimate Standard Errors*

Description

Calculate ordinary linear Regression and Estimate Standard Errors

Usage

```
mc.linreg(X, Y)
```

Arguments

X	measurement values of reference method.
Y	measurement values of test method.

Value

a list with elements

b0	intercept.
b1	slope.
se.b0	respective standard error of intercept.
se.b1	respective standard error of slope.
xw	average of reference method values.

References

Neter J., Wassermann W., Kunter M. Applied Statistical Models. Richard D. Irwing, INC., 1985.

---

mc.make.CIframe *Returns Results of Calculations in Matrix Form*

Description

Returns Results of Calculations in Matrix Form

Usage

```
mc.make.CIframe(b0, b1, se.b0, se.b1, CI.b0, CI.b1)
```

Arguments

b0	point estimate for intercept.
b1	point estimate for slope.
se.b0	standard error of intercept estimate.
se.b1	standard error of slope estimate.
CI.b0	numeric vector of length 2 - confidence interval for intercept.
CI.b1	numeric vector of length 2 - confidence interval for slope.

Value

2x4 matrix of estimates and confidence intervals for intercept and slope.

`mc.mdemingConstCV` *Calculate Weighted Deming Regression*

Description

Calculate weighted Deming regression with iterative algorithm suggested by Linnet. This algorithm is available only for positive values. But even in this case there is no guarantee that the algorithm always converges.

Usage

```
mc.mdemingConstCV(
  X,
  Y,
  error.ratio,
  iter.max = 30,
  threshold = 1e-06,
  kM = 1.345
)
```

Arguments

`X`	measurement values of reference method.
`Y`	measurement values of test method.
`error.ratio`	ratio between squared measurement errors of reference- and test method, necessary for Deming regression (Default is 1).
`iter.max`	maximal number of iterations.
`threshold`	threshold value
`kM`	Huber's k for the M weighting, default kM = 1.345

Value

a list with elements

`b0`	intercept.
`b1`	slope.
`xw`	average of reference method values.
`iter`	number of iterations.

References

Linnet K. Evaluation of Regression Procedures for Methods Comparison Studies. CLIN. CHEM. 39/3, 424-432 (1993).

Linnet K. Estimation of the Linear Relationship between the Measurements of two Methods with Proportional Errors. STATISTICS IN MEDICINE, Vol. 9, 1463-1473 (1990).

`mc.mmdemingConstCV` *Calculate Weighted Deming Regression*

Description

Calculate weighted Deming regression with iterative algorithm suggested by Linnet. This algorithm is available only for positive values. But even in this case there is no guarantee that the algorithm always converges.

Usage

```
mc.mmdemingConstCV(
  X,
  Y,
  error.ratio,
  iter.max = 120,
  threshold = 1e-06,
  tauMM = 4.685
)
```

Arguments

`X`	measurement values of reference method.
`Y`	measurement values of test method.
`error.ratio`	ratio between squared measurement errors of reference- and test method, necessary for Deming regression (Default is 1).
`iter.max`	maximal number of iterations.
`threshold`	threshold value.
`tauMM`	Tukey's tau for bisquare redescending weighting function, default tauMM = 4,685

Value

a list with elements

`b0`	intercept.
`b1`	slope.
`xw`	average of reference method values.
`iter`	number of iterations.

References

Linnet K. Evaluation of Regression Procedures for Methods Comparison Studies. CLIN. CHEM. 39/3, 424-432 (1993).

Linnet K. Estimation of the Linear Relationship between the Measurements of two Methods with Proportional Errors. STATISTICS IN MEDICINE, Vol. 9, 1463-1473 (1990).

`mc.mmNgdemingConstCV` *Calculate MM Deming Regression*

Description

Calculate MM Deming regression with iterative algorithm inspired on the work of Linnet. The algorithm uses bisquare redescending weights. For maximal stability and convergence the euclidean residuals are scaled in each iteration with a fresh calculated MAD instead of keeping the same MAD (assessed at the starting step) for the whole iteration. This algorithm is available only for positive values. But even in this case there is no guarantee that the algorithm always converges.

Usage

```
mc.mmNgdemingConstCV(
  X,
  Y,
  error.ratio,
  iter.max = 30,
  threshold = 1e-06,
  kM = 1.345,
  tauMM = 4.685,
  bdPoint = 0.5,
  priorSlope = 1,
  priorIntercept = 0
)
```

Arguments

`X`	measurement values of reference method.
`Y`	measurement values of test method.
`error.ratio`	ratio between squared measurement errors of reference- and test method, necessary for Deming regression (Default is 1).
`iter.max`	maximal number of iterations.
`threshold`	threshold value.
`kM`	Huber's k for the M weighting, default kM = 1.345
`tauMM`	Tukey's tau for bisquare redescending weighting function, default tauMM = 4,685
`bdPoint`	Proportion of data points selected for the highly robust M regression used for the determination of the starting parameters. Default 0.5
`priorSlope`	starting slope value for PiMMDeming, default priorSlope = 1
`priorIntercept`	starting intercept value for PiMMDeming, default priorIntercept = 0

Value

a list with elements

b0	intercept.
b1	slope.
xw	average of reference method values.
iter	number of iterations.

References

Linnet K. Evaluation of Regression Procedures for Methods Comparison Studies. CLIN. CHEM. 39/3, 424-432 (1993).

Linnet K. Estimation of the Linear Relationship between the Measurements of two Methods with Proportional Errors. STATISTICS IN MEDICINE, Vol. 9, 1463-1473 (1990).

---

mc.mmPidemingConstCV *Calculate MM Deming Regression*

Description

Calculate MM Deming regression with iterative algorithm inspired on the work of Linnet. The algorithm uses bisquare redescending weights. For maximal stability and convergence the euclidean residuals are scaled in each iteration with a fresh calculated MAD instead of keeping the same MAD (assessed at the starting step) for the whole iteration. This algorithm is available only for positive values. But even in this case there is no guarantee that the algorithm always converges.

Usage

```
mc.mmPidemingConstCV(
  X,
  Y,
  error.ratio,
  iter.max = 30,
  threshold = 1e-06,
  priorSlope = 1,
  priorIntercept = 0,
  tauMM = 4.685,
  kM = 1.345
)
```

Arguments

X	measurement values of reference method.
Y	measurement values of test method.
error.ratio	ratio between squared measurement errors of reference- and test method, necessary for Deming regression (Default is 1).

iter.max	maximal number of iterations.
threshold	threshold value.
priorSlope	starting slope value, default priorSlope = 1
priorIntercept	starting intercept value, default priorIntercept = 0
tauMM	Tukey's tau for bisquare redescending weighting function, default tauMM = 4,685
kM	description

Value

a list with elements

b0	intercept.
b1	slope.
xw	average of reference method values.
iter	number of iterations.

References

Linnet K. Evaluation of Regression Procedures for Methods Comparison Studies. CLIN. CHEM. 39/3, 424-432 (1993).

Linnet K. Estimation of the Linear Relationship between the Measurements of two Methods with Proportional Errors. STATISTICS IN MEDICINE, Vol. 9, 1463-1473 (1990).

---

mc.paba *Passing-Bablok Regression*

Description

Passing-Bablok Regression

Usage

```
mc.paba(
  angM = NULL,
  X,
  Y,
  alpha = 0.05,
  posCor = TRUE,
  calcCI = TRUE,
  slope.measure = c("radian", "tangent")
)
```

Arguments

angM	upper triangular matrix of slopes for all point combinations (optional). Slopes in radian.
X	measurement values of reference method
Y	measurement values of test method
alpha	numeric value specifying the 100(1-alpha)% confidence level
posCor	should algorithm assume positive correlation, i.e. symmetry around slope 1?
calcCI	should confidence intervals be computed?
slope.measure	angular measure of pairwise slopes (see mcreg for details). "radian" - for data sets with even sample numbers median slope is calculated as average of two central slope angles. "tangent" - for data sets with even sample numbers median slope is calculated as average of two central slopes (tan(angle)).

Value

Matrix of estimates and confidence intervals for intercept and slope. No standard errors provided by this algorithm.

---

mc.paba.LargeData *Passing-Bablok Regression for Large Datasets*

Description

This function represents an interface to a fast C-implementation of an adaption of the Passing-Bablok algorithm for large datasets. Instead of building the complete matrix of pair-wise slope values, a pre-defined binning of slope-values is used (Default NBins=1e06). This reduces the required memory dramatically and speeds up the computation.

Usage

```
mc.paba.LargeData(
  X,
  Y,
  NBins = 1e+06,
  alpha = 0.05,
  posCor = TRUE,
  calcCI = TRUE,
  slope.measure = c("radian", "tangent")
)
```

Arguments

X	(numeric) vector containing measurement values of reference method
Y	(numeric) vector containing measurement values of test method
NBins	(integer) value specifying the number of bins used to classify slope-values
alpha	(numeric) value specifying the 100(1-alpha)% confidence level for confidence intervals
posCor	(logical) should algorithm assume positive correlation, i.e. symmetry around slope 1?
calcCI	(logical) should confidence intervals be computed?
slope.measure	angular measure of pairwise slopes (see mcreg for details). "radian" - for data sets with even sample numbers median slope is calculated as average of two central slope angles. "tangent" - for data sets with even sample numbers median slope is calculated as average of two central slopes (tan(angle)).

Value

Matrix of estimates and confidence intervals for intercept and slope. No standard errors provided by this algorithm.

Author(s)

Andre Schuetzenmeister <andre.schuetzenmeister@roche.com> (partly re-using code of function 'mc.paba')

Examples

```
 library("mcrPioda")
 data(creatinine,package="mcrPioda")

# remove any NAs
crea <- na.omit(creatinine)

# call the approximative Passing-Bablok algorithm (Default NBins=1e06)
res1 <- mcreg(x=crea[,1], y=crea[,2], method.reg="PaBaLarge", method.ci="analytical")
getCoefficients(res1)

# now increase the number of bins and see whether this makes a difference
res2 <- mcreg(x=crea[,1], y=crea[,2], method.reg="PaBaLarge", method.ci="analytical", NBins=1e07)
getCoefficients(res2)
getCoefficients(res1)-getCoefficients(res2)
```

mc.PBequi *Equivariant Passing-Bablok Regression*

Description

This is an implementation of the equivariant Passing-Bablok regression.

Usage

```
mc.PBequi(
  X,
  Y,
  alpha = 0.05,
  slope.measure = c("radian", "tangent"),
  method.reg = c("PBequi", "TS"),
  extended.output = FALSE,
  calcCI = TRUE,
  methodlarge = TRUE
)
```

Arguments

X	measurement values of reference method
Y	measurement values of test method
alpha	numeric value specifying the 100(1-alpha)% confidence level
slope.measure	angular measure of pairwise slopes (see mcreg for details). "radian" - for data sets with even sample numbers median slope is calculated as average of two central slope angles. "tangent" - for data sets with even sample numbers median slope is calculated as average of two central slopes (tan(angle)).
method.reg	"PBequi" equivariant Passing-Bablok regression; "TS" Theil-Sen regression
extended.output	boolean. If TRUE, several intermediate results are returned
calcCI	boolean. If TRUE, sd of intercept and slope as well as xw are calculated
methodlarge	If TRUE (default), quasilinear method is used, if FALSE, quadratic method is used

Value

a list with elements.

b0	intercept.
b1	slope.
se.b0	respective standard error of intercept.

se.b1	respective standard error of slope.
xw	weighted average of reference method values.
weight	dummy values, only returned it extended.output=FALSE.
sez	variance of intercept for fixed slope (extended.output=TRUE, only).
vartau	variance of Kendall's tau (extended.output=TRUE, only).
covtx	covariance of tau and zeta (extended.output=TRUE, only).
x0	"center of gravity" of x (extended.output=TRUE, only).
taui	"Inversion vector; Indicator of influence"

`mc.wdemingConstCV` *Calculate Weighted Deming Regression*

Description

Calculate weighted Deming regression with iterative algorithm suggested by Linnet. This algorithm is available only for positive values. But even in this case there is no guarantee that the algorithm always converges.

Usage

```
mc.wdemingConstCV(X, Y, error.ratio, iter.max = 30, threshold = 1e-06)
```

Arguments

X	measurement values of reference method.
Y	measurement values of test method.
error.ratio	ratio between squared measurement errors of reference- and test method, necessary for Deming regression (Default is 1).
iter.max	maximal number of iterations.
threshold	threshold value.

Value

a list with elements

b0	intercept.
b1	slope.
xw	average of reference method values.
iter	number of iterations.

References

Linnet K. Evaluation of Regression Procedures for Methods Comparison Studies. CLIN. CHEM. 39/3, 424-432 (1993).

Linnet K. Estimation of the Linear Relationship between the Measurements of two Methods with Proportional Errors. STATISTICS IN MEDICINE, Vol. 9, 1463-1473 (1990).

mc.wlinreg — *Calculate Weighted Ordinary Linear Regression and Estimate Standard Errors*

Description

The weights of regression are taken as reverse squared values of the reference method, that's why it is impossible to achieve the calculations for zero values.

Usage

```
mc.wlinreg(X, Y)
```

Arguments

X	measurement values of reference method.
Y	measurement values of test method.

Value

a list with elements.

b0	intercept.
b1	slope.
se.b0	respective standard error of intercept.
se.b1	respective standard error of slope.
xw	weighted average of reference method values.

References

Neter J., Wassermann W., Kunter M. Applied Statistical Models. Richard D. Irwing, INC., 1985.

mcreg — *Comparison of Two Measurement Methods Using Regression Analysis*

Description

mcreg is used to compare two measurement methods by means of regression analysis. Available methods comprise ordinary and weighted linear regression, Deming and weighted Deming regression and Passing-Bablok regression. Point estimates of regression parameters are computed together with their standard errors and confidence intervals.

Usage

```
mcreg(
  x,
  y = NULL,
  error.ratio = 1,
  alpha = 0.05,
  mref.name = NULL,
  mtest.name = NULL,
  sample.names = NULL,
 method.reg = c("PaBa", "LinReg", "WLinReg", "Deming", "WDeming", "PaBaLarge", "PBequi",
    "TS", "MDeming", "MMDeming", "NgMMDeming", "PiMMDeming"),
  method.ci = c("bootstrap", "jackknife", "analytical", "nestedbootstrap"),
  method.bootstrap.ci = c("quantile", "Student", "BCa", "tBoot"),
  nsamples = 999,
  nnested = 25,
  rng.seed = NULL,
  rng.kind = "Mersenne-Twister",
  iter.max = 30,
  threshold = 1e-06,
  na.rm = FALSE,
  NBins = 1e+06,
  kM = 1.345,
  tauMM = 4.685,
  priorSlope = 1,
  priorIntercept = 0,
  bdPoint = 0.5,
  slope.measure = c("radian", "tangent"),
  methodlarge = TRUE
)
```

Arguments

x	measurement values of reference method, or two column matrix.
y	measurement values of test method.
error.ratio	ratio between squared measurement errors of reference and test method, necessary for Deming regression (Default 1).
alpha	value specifying the 100(1-alpha)% confidence level for confidence intervals (Default is 0.05).
mref.name	name of reference method (Default "Method1").
mtest.name	name of test Method (Default "Method2").
sample.names	names of cases (Default "S##").
method.reg	regression method. It is possible to choose between five regression methods: "LinReg" - ordinary least square regression. "WLinReg" - weighted ordinary least square regression. "Deming" - Deming regression. "WDeming" - weighted Deming regression.

"MDeming" - Huber M-Deming regression.
"MMDeming" - Huber MM-Deming regression.
"NgMMDeming" - new generation MM-Deming regression.
"PiMMDeming" - Prior informed MM-Deming regression.
"TS" - Theil-Sen regression.
"PBequi" - equivariant Passing-Bablok regression.
"PaBa" - Passing-Bablok regression.
"PaBaLarge" - approximative Passing-Bablok regression for large datasets, operating on NBins classes of constant slope angle which each slope is classified to instead of building the complete triangular matrix of all N*N/2 slopes.

method.ci — method of confidence interval calculation. The function contains four basic methods for calculation of confidence intervals for regression coefficients. "analytical" - with parametric method.
"jackknife" - with leave one out resampling.
"bootstrap" - with ordinary non-parametric bootstrap resampling.
"nested bootstrap" - with ordinary non-parametric bootstrap resampling.

method.bootstrap.ci — bootstrap based confidence interval estimation method.

nsamples — number of bootstrap samples.

nnested — number of nested bootstrap samples.

rng.seed — integer number that sets the random number generator seed for bootstrap sampling. If set to NULL currently in the R session used RNG setting will be used.

rng.kind — type of random number generator for bootstrap sampling. Only used when rng.seed is specified, see set.seed for details.

iter.max — maximum number of iterations for weighted Deming iterative algorithm.

threshold — numerical tolerance for weighted Deming iterative algorithm convergence.

na.rm — remove measurement pairs that contain missing values (Default is FALSE).

NBins — number of bins used when 'reg.method="PaBaLarge"' to classify each slope in one of 'NBins' bins covering the range of all slopes

kM — Huber's k for the M weighting, default kM = 1.345

tauMM — Tukey's tau for bisquare redescending weighting function, default tauMM = 4,685

priorSlope — starting slope value for PiMMDeming, default priorSlope = 1

priorIntercept — starting intercept value for PiMMDeming, default priorIntercept = 0

bdPoint — Proportion of data points selected for the highly robust M regression used for the determination of the starting parameters. Default 0.5

slope.measure — angular measure of pairwise slopes used for exact PaBa regression (see below for details).
"radian" - for data sets with even sample numbers median slope is calculated as average of two central slope angles.
"tangent" - for data sets with even sample numbers median slope is calculated as average of two central slopes (tan(angle)).

`methodlarge` Boolean. This parameter applies only to regmethod="PBequi" and "TS". If TRUE, a quasilinear algorithm is used. If FALSE, a quadratic algorithm is used which is faster for less than several hundred data pairs.

Details

The regression analysis yields regression coefficients 'Intercept' and 'Slope' of the regression $Testmethod = Intercept + Slope * Referencemethod$. There are methods for computing the systematical bias between reference and test method at a decision point Xc, $Bias(Xc) = Intercept + (Slope - 1) * Xc$, accompanied by its corresponding standard error and confidence interval. One can use plotting method `plotBias` for a comprehensive view of the systematical bias.

Weighted regression for heteroscedastic data is available for linear and Deming regression and implemented as a data point weighting with the inverted squared value of the reference method. Therefore calculation of weighted regression (linear and Deming) is available only for positive values (>0). Passing-Bablok regression is only available for non-negative values (>=0).

Confidence intervals for regression parameters and bias estimates are calculated either by using analytical methods or by means of resampling methods ("jackknife", "bootstrap", "nested bootstrap"). An analytical method is available for all types of regression except for weighted Deming. For Passing-Bablok regression the option "analytical" calculates confidence intervals for the regression parameters according to the non-parametric approach given in the original reference.

The "jackknife" (or leave one out resampling) method was suggested by Linnet for calculating confidence intervals of regression parameters of Deming and weighted Deming regression. It is possible to calculate jackknife confidence intervals for all types of regression. Note that we do not recommend this method for Passing-Bablok since it has a tendency of underestimating the variability (jackknife is known to yield incorrect estimates for errors of quantiles).

The bootstrap method requires additionally choosing a value for `method.bootstrap.ci`. If bootstrap is the method of choice, "BCa", t-bootstrap ("tBoot") and simple "quantile" confidence intervals are recommended (See Efron B. and Tibshirani R.J.(1993),Carpenter J., Bithell J. (2000)). The "nestedbootstrap" method can be very time-consuming but is necessary for calculating t-bootstrap confidence intervals for weighted Deming or Passing-Bablok regression. For these regression methods there are no analytical solutions for computing standard errors, which therefore have to be obtained by nested bootstrapping.

Note that estimating resampling based confidence intervals for Passing–Bablok regressions can take very long time for larger data sets due to the high computational complexity of the algorithm. To mitigate this drawback an adaption of the Passing-Bablok algorithm has been implemented (`"PaBaLarge"`), which yields approximative results. This approach does not build the complete upper triangular matrix of all 'n*(n-1)/2' slopes. It subdivides the range of slopes into 'NBins' classes, and sorts each slope into one of these bins. The remaining steps are the same as for the exact `"PaBa"` algorithm, except that these are performed on the binned slopes instead of operating on the matrix of slopes.

Our implementation of exact Passing-Bablok regression (`"PaBa"`) provides two alternative metrics for regression slopes which can result in different regression estimates. As a robust regression method PaBa is essentially invariant to the parameterization of regression slopes, however in the case of an even number of all pairwise slopes the two central slopes are averaged to estimate the final regression slope. In this situation using an angle based metric (`slope.measure="radian"`) will result in a regression estimate that is geometrically centered between the two central slopes, whereas the tangent measure (`slope.measure="tangent"`) proposed in Passing and Bablok (1983)

will be geometrically biased towards a higher slope. See below for a pathological example. Note that the difference between the two measures is negligible for data sets with reasonable sample size (N>20) and correlation.

Equivariant Passing-Bablok regression as proposed by Bablok et al. (1988) (see also Dufey 2020) is not bound to slopes near 1 and therefore not only applicable for method comparison but also for method transformation, i.e., when two methods yield results on a different scale. Like ordinary Passing-Bablok regression, the method is robust. This method should be preferred over the older "PaBa" and "PaBalarge" algorithms. Both slope measures "radian" and "tangent" are available as are methods for the determination of confidence intervals -analytical and bootstrap. By default (methodlarge=TRUE), a modified algorithm (Dillencourt et al., 1992) is used which scales quasilinearly and requires little memory. Alternatively (methodlarge=F), a simpler implementation which scales quadratically and is more memory intensive may be called. While point estimates coincide for both implementations, analytic confidence intervals differ slightly. Same holds true for the Theil-Sen estimator, which is a robust alternative to linear regression. Like linear regression, it assumes that x-values are error free.

Value

"MCResult" object containing regression results. The function `getCoefficients` or `printSummary` can be used to obtain or print a summary of the results. The function `getData` allows to see the original data. An S4 object of class "MCResult" containing at least the following slots:

`data`	measurement data in wide format, one pair of observations per sample. Includes samples ID, reference measurement, test measurement.
`para`	numeric matrix with estimates for slope and intercept, corresponding standard deviations and confidence intervals.
`mnames`	character vector of length two containing names of analytical methods.
`regmeth`	type of regression type used for parameter estimation.
`cimeth`	method used for calculation of confidence intervals.
`error.ratio`	ratio between squared measurement errors of reference and test method, necessary for Deming regression.
`alpha`	confidence level using for calculation of confidence intervals.

Author(s)

Ekaterina Manuilova <ekaterina.manuilova@roche.com>, Andre Schuetzenmeister <andre.schuetzenmeister@roche.
Fabian Model <fabian.model@roche.com>, Sergej Potapov <sergej.potapov@roche.com>, Florian Dufey <florian.dufey@roche.com>, Jakob Raymaekers <jakob.raymaekers@kuleuven.be>

References

Bland, J. M., Altman, D. G. (1986) Statistical methods for assessing agreement between two methods of clinical measurement. *Lancet*, **i:** 307–310.

Linnet, K. (1993) Evaluation of Regression Procedures for Methods Comparison Studies. *CLIN. CHEM.* **39/3**, 424–432.

Linnet, K. (1990) Estimation of the Linear Relationship between the Measurements of two Methods with Proportional Errors. *Statistics in Medicine*, Vol. **9**, 1463–1473.

Neter, J., Wassermann, W., Kunter, M. (1985) *Applied Statistical Models.* Richard D. Irwing, INC.

Looney, S. W. (2010) Statistical Methods for Assessing Biomarkers. *Methods in Molecular Biology*, vol. **184**: *Biostatistical Methods*. Human Press INC.

Passing, H., Bablok, W. (1983) A new biometrical procedure for testing the equality of measurements from two different analytical methods. Application of linear regression procedures for method comparison studies in clinical chemistry, Part I. *J Clin Chem Clin Biochem*. Nov; **21(11)**:709–20.

Bablok, W., Passing, H., Bender, R., & Schneider, B. (1988) A general regression procedure for method transformation. Application of linear regression procedures for method comparison studies in clinical chemistry, Part III. *Clinical Chemistry and Laboratory Medicine*, **26(11)**: 783–790.

Dillencourt, M. B., Mount, D. M., & Netanyahu, N. S. (1992) A randomized algorithm for slope selection. *International Journal of Computational Geometry & Applications*, **2(01)**: 1–27.

Dufey, F. (2020) Derivation of Passing-Bablok regression from Kendall's tau. *The International Journal of Biostatistics*, **16(2)**: 20190157. https://doi.org/10.1515/ijb-2019-0157

Raymaekers, J., Dufey, F. (2022) Equivariant Passing-Bablok regression in quasilinear time. *arXiv preprint arXiv:2202.08060*. https://doi.org/10.48550/arXiv.2202.08060

Efron, B., Tibshirani, R.J. (1993) *An Introduction to the Bootstrap*. Chapman and Hall.

Carpenter, J., Bithell, J. (2000) Bootstrap confidence intervals: when, which, what? A practical guide for medical statisticians. *Stat Med*, **19 (9)**, 1141–1164.

CLSI EP9-A2. Method Comparison and Bias Estimation Using Patient Samples; Approved Guideline.

See Also

plotDifference, plot.mcr, getResiduals, plotResiduals, calcResponse, calcBias, plotBias, compareFit

Examples

```
library("mcrPioda")
data(creatinine,package="mcrPioda")
x <- creatinine$serum.crea
y <- creatinine$plasma.crea
# Deming regression fit.
# The confidence intercals for regression coefficients
# are calculated with analytical method
model1<- mcreg(x,y,error.ratio=1,method.reg="Deming", method.ci="analytical",
               mref.name = "serum.crea", mtest.name = "plasma.crea", na.rm=TRUE)
# Results
printSummary(model1)
getCoefficients(model1)
plot(model1)
# Deming regression fit.
# The confidence intervals for regression coefficients
# are calculated with bootstrap (BCa) method
model2<- mcreg(x,y,error.ratio=1,method.reg="Deming",
               method.ci="bootstrap", method.bootstrap.ci = "BCa",
               mref.name = "serum.crea", mtest.name = "plasma.crea", na.rm=TRUE)
```

```
compareFit(model1, model2)

## Pathological example of Passing-Bablok regression where measure for slope angle matters
x1 <- 1:10; y1 <- 0.5*x1; x <- c(x1,y1); y <- c(y1,x1)
m1 <- mcreg(x,y,method.reg="PaBa",method.ci="analytical",slope.measure="radian",
            mref.name="X",mtest.name="Y")
m2 <- mcreg(x,y,method.reg="PaBa",method.ci="analytical",slope.measure="tangent",
            mref.name="X",mtest.name="Y")
plot(m1, add.legend=FALSE,identity=FALSE,
   main="Radian vs. tangent slope measures in Passing-Bablok regression\n(pathological example)",
     ci.area=FALSE,add.cor=FALSE)
plot(m2, ci.area=FALSE,reg.col="darkgreen",reg.lty=2,identity=FALSE,add.legend=FALSE,
     draw.points=FALSE,add=TRUE,add.cor=FALSE)
includeLegend(place="topleft",models=list(m1,m2),model.names=c("PaBa Radian","PaBa Tangent"),
              colors=c("darkblue","darkgreen"),lty=c(1,2),design="1",digits=2)
```

`MCResult-class` *Class* `"MCResult"`

Description

Result of a method comparison.

Objects from the Class

Object is typically created by a call to function `mcreg`. Object can be directly constructed by calling `newMCResult` or `new("MCResult", data, para, mnames, regmeth, cimeth, error.ratio, alpha, weight)`.

Slots

`data`: Object of class `"data.frame"` ~~

`para`: Object of class `"matrix"` ~~

`mnames`: Object of class `"character"` ~~

`regmeth`: Object of class `"character"` ~~

`cimeth`: Object of class `"character"` ~~

`error.ratio`: Object of class `"numeric"` ~~

`alpha`: Object of class `"numeric"` ~~

`weight`: Object of class `"numeric"` ~~

Methods

calcBias `signature(.Object = "MCResult")`: ...

calcCUSUM `signature(.Object = "MCResult")`: ...

calcResponse `signature(.Object = "MCResult")`: ...

getCoefficients `signature(.Object = "MCResult")`: ...

coef `signature(.Object = "MCResult")`: ...

getData `signature(.Object = "MCResult")`: ...

getErrorRatio `signature(.Object = "MCResult")`: ...

getRegmethod `signature(.Object = "MCResult")`: ...

getResiduals `signature(.Object = "MCResult")`: ...

getWeights `signature(.Object = "MCResult")`: ...

plot `signature(x = "MCResult")`: ...

plotBias `signature(x = "MCResult")`: ...

plotDifference `signature(.Object = "MCResult")`: ...

plotResiduals `signature(.Object = "MCResult")`: ...

printSummary `signature(.Object = "MCResult")`: ...

summary `signature(.Object = "MCResult")`: ...

Author(s)

Ekaterina Manuilova <ekaterina.manuilova@roche.com>, Andre Schuetzenmeister <andre.schuetzenmeister@roche.c
Fabian Model <fabian.model@roche.com> Sergej Potapov <sergej.potapov@roche.com> Giorgio Pioda <gfwp@ticino.com>

Examples

```
showClass("MCResult")
```

MCResult.calcBias *Systematical Bias Between Reference Method and Test Method*

Description

Calculate systematical bias between reference and test methods at the decision point Xc as $Bias(Xc) = Intercept + (Slope - 1) * Xc$ with corresponding confidence intervals.

Usage

```
MCResult.calcBias(
  .Object,
  x.levels,
  type = c("absolute", "proportional"),
  percent = TRUE,
  alpha = 0.05,
  ...
)
```

Arguments

.Object	object of class "MCResult".
x.levels	a numeric vector with decision points for which bias should be calculated.
type	One can choose between absolute (default) and proportional bias (Bias(Xc)/Xc).
percent	logical value. If percent = TRUE the proportional bias will be calculated in percent.
alpha	numeric value specifying the 100(1-alpha)% confidence level of the confidence interval (Default is 0.05).
...	further parameters

Value

response and corresponding confidence interval for each decision point from x.levels.

See Also

plotBias

Examples

```
#library("mcr")
data(creatinine,package="mcrPioda")
x <- creatinine$serum.crea
y <- creatinine$plasma.crea

# Deming regression fit.
# The confidence intervals for regression coefficients
# are calculated with analytical method
model <- mcreg( x,y,error.ratio = 1,method.reg = "Deming", method.ci = "analytical",

                mref.name = "serum.crea", mtest.name = "plasma.crea", na.rm=TRUE )
# Now we calculate the systematical bias
# between the testmethod and the reference method
# at the medical decision points 1, 2 and 3

calcBias( model, x.levels = c(1,2,3))
calcBias( model, x.levels = c(1,2,3), type = "proportional")
calcBias( model, x.levels = c(1,2,3), type = "proportional", percent = FALSE)
```

MCResult.calcCUSUM *Calculate CUSUM Statistics According to Passing & Bablok (1983)*

Description

Calculate CUSUM Statistics According to Passing & Bablok (1983)

Usage

```
MCResult.calcCUSUM(.Object)
```

Arguments

.Object	object of class "MCResult".

Value

A list containing the following elements:

nPos	sum of positive residuals
nNeg	sum of negative residuals
cusum	a cumulative sum of vector with scores ri for each point, sorted increasing by distance of points to regression line.
max.cumsum	Test statistics of linearity test

References

Passing, H., Bablok, W. (1983) A new biometrical procedure for testing the equality of measurements from two different analytical methods. Application of linear regression procedures for method comparison studies in clinical chemistry, Part I. *J Clin Chem Clin Biochem*. Nov; **21(11)**:709–20.

MCResult.calcPaBaTiesRatio

Calculate PaBa Ties Ratio.

Description

This function computes the ratio of slopes ties values in the classic pairwise PaBa slope calculation. A ratio higher than approx 0.05 may suggest a potential bias risk. In this case data precision is too low for correct PaBa estimation. The function is written in R and can be slow with large data set.

Usage

```
MCResult.calcPaBaTiesRatio(.Object)
```

Arguments

.Object	object of class "MCResult".

Value

PaBa ties ratio (ties slopes vs total slopes)

`MCResult.calcResponse` *Calculate Response with Confidence Interval.*

Description

Calculate Response $Intercept + Slope * Refrencemethod$ with Corresponding Confidence Interval

Usage

```
MCResult.calcResponse(.Object, x.levels, alpha, ...)
```

Arguments

`.Object`	object of class "MCResult".
`x.levels`	a numeric vector with points for which response should be calculated.
`alpha`	numeric value specifying the 100(1-`alpha`)% confidence level of the confidence interval (Default is 0.05).
`...`	further parameters

Value

response and corresponding confidence interval for each point in vector `x.levels`.

See Also

`calcBias`

Examples

```
#library("mcr")
data(creatinine,package="mcrPioda")
x <- creatinine$serum.crea
y <- creatinine$plasma.crea
# Deming regression fit.
# The confidence intercals for regression coefficients
# are calculated with analytical method
model <- mcreg( x,y,error.ratio=1,method.reg="Deming", method.ci="analytical",
                mref.name = "serum.crea", mtest.name = "plasma.crea", na.rm=TRUE )
calcResponse(model, x.levels=c(1,2,3))
```

MCResult.getCoefficients *Get Regression Coefficients*

Description

Get Regression Coefficients

Usage

```
MCResult.getCoefficients(.Object)
```

Arguments

.Object	object of class "MCResult".

Value

Regression parameters in matrix form. Rows: Intercept, Slope. Cols: EST, SE, LCI, UCI.

MCResult.getData *Get Data*

Description

Get Data

Usage

```
MCResult.getData(.Object)
```

Arguments

.Object	object of class "MCResult".

Value

Measurement data in matrix format. First column contains reference method (X), second column contains test method (Y).

MCResult.getErrorRatio *Get Error Ratio*

Description

Get Error Ratio

Usage

```
MCResult.getErrorRatio(.Object)
```

Arguments

.Object	Object of class "MCResult"

Value

Error ratio. Only relevant for Deming type regressions.

MCResult.getFitted *Get Fitted Values.*

Description

This function computes fitted values for a 'MCResult'-object. Depending on the regression method and the error ratio, a projection onto the regression line is performed accordingly. For each point (x_i; y_i) i=1,...,n the projected point(x_hat_i; y_hat_i) is computed.

Usage

```
MCResult.getFitted(.Object)
```

Arguments

.Object	object of class "MCResult".

Value

fitted values as data frame.

See Also

plotResiduals getResiduals

MCResult.getRegmethod *Get Regression Method*

Description

Get Regression Method

Usage

```
MCResult.getRegmethod(.Object)
```

Arguments

.Object	object of class "MCResult".

Value

Name of the statistical method used for the regression analysis.

MCResult.getResiduals *Get Regression Residuals*

Description

This function returns residuals in x-direction (x-xhat), in y-direction(y-yhat) and optimized residuals. The optimized residuals correspond to distances between data points and the regression line which were optimized for regression coefficients estimation. In case of Passing-Bablok Regression orthogonal residuals will be returned as optimized residuals . The residuals in x-direction are interesting for regression types which assume errors in both variables (Deming, weighted Deming, Passing-Bablok), particularily for checking of model assumptions.

Usage

```
MCResult.getResiduals(.Object)
```

Arguments

.Object	object of class "MCResult".

Value

residuals as data frame.

See Also

plotResiduals

MCResult.getWeights *Get Weights of Data Points*

Description

Get Weights of Data Points

Usage

```
MCResult.getWeights(.Object)
```

Arguments

.Object	Object of class "MCResult"

Value

Weights of data points.

MCResult.initialize *MCResult Object Initialization*

Description

MCResult Object Initialization

Usage

```
MCResult.initialize(
  .Object,
  data = data.frame(X = NA, Y = NA),
  para = matrix(NA, ncol = 4, nrow = 2),
  mnames = c("unknown", "unknown"),
  regmeth = "unknown",
  cimeth = "unknown",
  error.ratio = 0,
  alpha = 0.05,
  weight = 1
)
```

Arguments

.Object	object of class "MCResult"
data	measurement data in matrix format. First column reference method (x), second column test method (y).
para	regression parameters in matrix form. Rows: Intercept, Slope. Cols: EST, SE, LCI, UCI.
mnames	names of reference and test method.
regmeth	name of statistical method used for regression.
cimeth	name of statistical method used for computing confidence intervals.
error.ratio	ratio between standard deviation of reference and test method.
alpha	numeric value specifying the 100(1-alpha)% confidence level of confidence intervals (Default is 0.05).
weight	weights to be used for observations

Value

MCResult object with initialized parameter.

MCResult.plot *Scatter Plot Method X vs. Method Y*

Description

Plot method X (reference) vs. method Y (test) with (optional) line of identity, regression line and confidence bounds for response.

Usage

```
MCResult.plot(
  x,
  alpha = 0.05,
  xn = 20,
  equal.axis = FALSE,
  xlim = NULL,
  ylim = NULL,
  xaxp = NULL,
  yaxp = NULL,
  x.lab = x@mnames[1],
  y.lab = x@mnames[2],
  add = FALSE,
  draw.points = TRUE,
  points.col = "black",
  points.pch = 1,
  points.cex = 0.8,
```

```
  reg = TRUE,
  reg.col = NULL,
  reg.lty = 1,
  reg.lwd = 2,
  identity = TRUE,
  identity.col = NULL,
  identity.lty = 2,
  identity.lwd = 1,
  ci.area = TRUE,
  ci.area.col = NULL,
  ci.border = FALSE,
  ci.border.col = NULL,
  ci.border.lty = 2,
  ci.border.lwd = 1,
  add.legend = TRUE,
  legend.place = c("topleft", "topright", "bottomleft", "bottomright"),
  main = NULL,
  sub = NULL,
  add.cor = TRUE,
  cor.method = c("pearson", "kendall", "spearman"),
  add.grid = TRUE,
  digits = list(coef = 2, cor = 3),
  ...
)
```

Arguments

x	object of class "MCResult".
alpha	numeric value specifying the 100(1-alpha)% confidence bounds.
xn	number of points (default 20) for calculation of confidence bounds.
equal.axis	logical value. If equal.axis=TRUE x-axis will be equal to y-axis.
xlim	limits of the x-axis. If xlim=NULL the x-limits will be calculated automatically.
ylim	limits of the y-axis. If ylim=NULL the y-limits will be calculated automatically.
xaxp	ticks of the x-axis. If xaxp=NULL the x-ticks will be calculated automatically.
yaxp	ticks of the y-axis. If yaxp=NULL the y-ticks will be calculated automatically.
x.lab	label of x-axis. Default is the name of reference method.
y.lab	label of y-axis. Default is the name of test method.
add	logical value. If add=TRUE, the plot will be drawn in current graphical window.
draw.points	logical value. If draw.points=TRUE, the data points will be drawn.
points.col	Color of data points.
points.pch	Type of data points (see par()).
points.cex	Size of data points (see par()).
reg	Logical value. If reg=TRUE, the regression line will be drawn.
reg.col	Color of regression line.

reg.lty	Type of regression line.
reg.lwd	The width of regression line.
identity	logical value. If identity=TRUE the identity line will be drawn.
identity.col	The color of identity line.
identity.lty	The type of identity line.
identity.lwd	the width of identity line.
ci.area	logical value. If ci.area=TRUE (default) the confidence area will be drawn.
ci.area.col	the color of confidence area.
ci.border	logical value. If ci.border=TRUE the confidence limits will be drawn.
ci.border.col	The color of confidence limits.
ci.border.lty	The line type of confidence limits.
ci.border.lwd	The line width of confidence limits.
add.legend	logical value. If add.legend=FALSE the plot will not have any legend.
legend.place	The position of legend: "topleft","topright","bottomleft","bottomright".
main	String value. The main title of plot. If main=NULL it will include regression name.
sub	String value. The subtitle of plot. If sub=NULL and ci.border=TRUE or ci.area=TRUE it will include the art of confidence bounds calculation.
add.cor	Logical value. If add.cor=TRUE the correlation coefficient will be shown.
cor.method	a character string indicating which correlation coefficient is to be computed. One of "pearson" (default), "kendall", or "spearman", can be abbreviated.
add.grid	Logical value. If add.grid=TRUE (default) the gridlines will be drawn.
digits	list with the number of digits for the regression equation and the correlation coefficient.
...	further graphical parameters

Value

No return value, instead a plot is generated

See Also

plotBias, plotResiduals, plotDifference, compareFit,includeLegend

Examples

```
library(mcrPioda)
data(creatinine,package="mcrPioda")
creatinine <- creatinine[complete.cases(creatinine),]
 x <- creatinine$serum.crea
 y <- creatinine$plasma.crea

 m1 <- mcreg(x,y,method.reg="Deming",  mref.name="serum.crea",
                                       mtest.name="plasma.crea", na.rm=TRUE)
```

```
m2 <- mcreg(x,y,method.reg="WDeming", method.ci="jackknife",
                                      mref.name="serum.crea",
                                      mtest.name="plasma.crea", na.rm=TRUE)

plot(m1,  xlim=c(0.5,3),ylim=c(0.5,3), add.legend=FALSE,
                         main="Deming vs. weighted Deming regression",
                         points.pch=19,ci.area=TRUE, ci.area.col=grey(0.9),
                         identity=FALSE, add.grid=FALSE, sub="")
plot(m2, ci.area=FALSE, ci.border=TRUE, ci.border.col="red3",
                         reg.col="red3", add.legend=FALSE,
                         draw.points=FALSE,add=TRUE)

includeLegend(place="topleft",models=list(m1,m2),
                         colors=c("darkblue","red"), design="1", digits=2)
```

MCResult.plotBias *Plot Estimated Systematical Bias with Confidence Bounds*

Description

This function plots the estimated systematical bias $(Intercept + Slope * Refrencemethod) - Referencemethod$ with confidence bounds, covering the whole range of reference method X or only part of it.

Usage

```
MCResult.plotBias(
  x,
  xn = 100,
  alpha = 0.05,
  add = FALSE,
  prop = FALSE,
  xlim = NULL,
  ylim = NULL,
  bias = TRUE,
  bias.lty = 1,
  bias.lwd = 2,
  bias.col = NULL,
  ci.area = TRUE,
  ci.area.col = NULL,
  ci.border = FALSE,
  ci.border.col = NULL,
  ci.border.lwd = 1,
  ci.border.lty = 2,
  zeroline = TRUE,
  zeroline.col = NULL,
  zeroline.lty = 2,
  zeroline.lwd = 1,
```

```
  main = NULL,
  sub = NULL,
  add.grid = TRUE,
  xlab = NULL,
  ylab = NULL,
  cut.point = NULL,
  cut.point.col = "red",
  cut.point.lwd = 2,
  cut.point.lty = 1,
  ...
)
```

Arguments

x	object of class "MCResult".
xn	# number of points for drawing of confidence bounds/area.
alpha	numeric value specifying the 100(1-alpha)% confidence level of confidence intervals (Default is 0.05).
add	logical value. If add=TRUE, the graphic will be drawn in current graphical window.
prop	a logical value. If prop=TRUE the proportional bias $\%bias(Xc) = [Intercept + (Slope - 1) * Xc]/Xc$ will be drawn.
xlim	limits of the x-axis. If xlim=NULL the x-limits will be calculated automatically.
ylim	limits of the y-axis. If ylim=NULL the y-limits will be calculated automatically.
bias	logical value. If identity=TRUE the bias line will be drawn. If ci.bounds=FALSE and ci.area=FALSE the bias line will be drawn always.
bias.lty	type of the bias line.
bias.lwd	width of the bias line.
bias.col	color of the bias line.
ci.area	logical value. If ci.area=TRUE (default) the confidence area will be drawn.
ci.area.col	color of the confidence area.
ci.border	logical value. If ci.border=TRUE the confidence limits will be drawn.
ci.border.col	color of the confidence limits.
ci.border.lwd	line width of confidence limits.
ci.border.lty	line type of confidence limits.
zeroline	logical value. If zeroline=TRUE the zero-line will be drawn.
zeroline.col	color of the zero-line.
zeroline.lty	type of the zero-line.
zeroline.lwd	width of the zero-line.
main	character string. The main title of plot. If main = NULL it will include regression name.

sub	character string. The subtitle of plot. If sub=NULL and ci.border=TRUE or ci.area=TRUE it will include the art of confidence bounds calculation.
add.grid	logical value. If grid=TRUE (default) the gridlines will be drawn.
xlab	label for the x-axis
ylab	label for the y-axis
cut.point	numeric value. Decision level of interest.
cut.point.col	color of the confidence bounds at the required decision level.
cut.point.lwd	line width of the confidence bounds at the required decision level.
cut.point.lty	line type of the confidence bounds at the required decision level.
...	further graphical parameters

Value

No return value, instead a plot is generated

See Also

calcBias, plot.mcr, plotResiduals, plotDifference, compareFit

Examples

```
#library("mcr")
data(creatinine,package="mcrPioda")

creatinine <- creatinine[complete.cases(creatinine),]
x <- creatinine$serum.crea
y <- creatinine$plasma.crea

# Calculation of models
m1 <- mcreg(x,y,method.reg="WDeming", method.ci="jackknife",
                mref.name="serum.crea",mtest.name="plasma.crea", na.rm=TRUE)
m2 <- mcreg(x,y,method.reg="WDeming", method.ci="bootstrap",
                method.bootstrap.ci="BCa",mref.name="serum.crea",
                mtest.name="plasma.crea", na.rm=TRUE)

# Grafical comparison of systematical Bias of two models
plotBias(m1, zeroline=TRUE,zeroline.col="black",zeroline.lty=1,
                ci.area=TRUE,ci.border=FALSE, ci.area.col=grey(0.9),
                main = "Bias between serum and plasma creatinine",
                sub="Comparison of Jackknife and BCa-Bootstrap confidence bounds ")
plotBias(m2, ci.area=FALSE, ci.border=TRUE, ci.border.lwd=2,
                ci.border.col="red",bias=FALSE ,add=TRUE)
includeLegend(place="topleft",models=list(m1,m2), lwd=c(10,2),
                lty=c(2,1),colors=c(grey(0.9),"red"), bias=TRUE,
                design="1", digits=4)

# Drawing of proportional bias
plotBias(m1, ci.area=FALSE, ci.border=TRUE)
plotBias(m1, ci.area=FALSE, ci.border=TRUE, prop=TRUE)
```

```
plotBias(m1, ci.area=FALSE, ci.border=TRUE, prop=TRUE, cut.point=0.6)
plotBias(m1, ci.area=FALSE, ci.border=TRUE, prop=TRUE, cut.point=0.6,
             xlim=c(0.4,0.8),cut.point.col="orange", cut.point.lwd=3, main ="")
```

`MCResult.plotDifference` *Bland-Altman Plot*

Description

Draw different Bland-Altman plot modifications (see parameter `plot.type`).

Usage

```
MCResult.plotDifference(
  .Object,
  xlab = NULL,
  ylab = NULL,
  ref.line = TRUE,
  ref.line.col = "black",
  ref.line.lty = 1,
  ref.line.lwd = 1,
  bias.line.lty = 1,
  bias.line.lwd = 1,
  bias.line.col = "red",
  bias.text.col = NULL,
  bias.text.cex = 0.8,
  loa.line.lty = 2,
  loa.line.lwd = 1,
  loa.line.col = "red",
  loa.text.col = NULL,
  plot.type = 3,
  main = NULL,
  cex = 0.8,
  digits = 2,
  add.grid = TRUE,
  ylim = NULL,
  ...
)
```

Arguments

`.Object`	object of class "MCResult".
`xlab`	label for the x-axis
`ylab`	label for the y-axis
`ref.line`	logical value. If `ref.line=TRUE` (default), the reference line will be drawn.

ref.line.col	reference line color.
ref.line.lty	reference line type.
ref.line.lwd	reference line width.
bias.line.lty	line type for estimated bias.
bias.line.lwd	line width for estimated bias.
bias.line.col	color of the line for estimated bias.
bias.text.col	color of the label for estimated bias (defaults to the same as bias.line.col.)
bias.text.cex	The magnification to be used for the label for estimated bias
loa.line.lty	line type for estimated limits of agreement.
loa.line.lwd	line width for estimated limits of agreement.
loa.line.col	color of the line for estimated limits of agreement.
loa.text.col	color of the label for estimated limits of agreement (defaults to the same as loa.line.col.)
plot.type	integer specifying a specific Bland-Altman plot modification (default is 3). Possible choices are: 1 - difference plot X vs. Y-X with null-line and mean plus confidence intervals. 2 - difference plot X vs. (Y-X)/X (relative differences) with null-line and mean. 3 - difference plot 0.5*(X+Y) vs. Y-X with null-line and mean plus confidence intervals. 4 - difference plot 0.5*(X+Y) vs. (Y-X)/X (relative differences) with null-line. 5 - difference plot rank(X) vs. Y-X with null-line and mean plus confidence intervals. 6 - difference plot rank(X) vs. (Y-X)/X (relative differences) with null-line and mean. 7 - difference plot sqrt(X*Y) vs. Y/X with null-line and mean plus confidence intervals calculated with help of log-transformation. 8 - difference plot 0.5*(X+Y) vs. (Y-X) / (0.5*(X+Y)) with null-line.
main	plot title.
cex	numeric value specifying the magnification factor used for points
digits	number of decimal places for the difference of means and standard deviation appearing in the plot.
add.grid	logical value. If add.grid=TRUE (Default) gridlines will be drawn.
ylim	limits for the y-axis
...	further graphical parameters

Value

No return value, instead a plot is generated

References

Bland, J. M., Altman, D. G. (1986) Statistical methods for assessing agreement between two methods of clinical measurement. *Lancet*, **i:** 307–310.

See Also

plot.mcr, plotResiduals, plotDifference, plotBias, compareFit

Examples

```
#library("mcr")
data(creatinine,package="mcrPioda")
x <- creatinine$serum.crea
y <- creatinine$plasma.crea

# Deming regression fit.
# The confidence intercals for regression coefficients
# are calculated with analytical method
model <- mcreg( x,y,error.ratio=1,method.reg="Deming", method.ci="analytical",
                mref.name = "serum.crea", mtest.name = "plasma.crea", na.rm=TRUE )

plotDifference( model ) # Default plot.type=3
plotDifference( model, plot.type=5)
plotDifference( model, plot.type=7, ref.line.lty=3, ref.line.col="green3" )
```

MCResult.plotResiduals

Plot Residuals of an MCResult Object

Description

Plot Residuals of an MCResult Object

Usage

```
MCResult.plotResiduals(
  .Object,
  res.type = c("optimized", "y", "x"),
  xaxis = c("yhat", "both", "xhat"),
  ref.line = TRUE,
  ref.line.col = "red",
  ref.line.lty = 2,
  ref.line.lwd = 1,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  add.grid = TRUE,
  ...
)
```

Arguments

.Object	object of type "MCResult".
res.type	If res.type="y" the difference between the test method and it's prediction will be drawn. If res.type="x" the reference method and it's prediction will be drawn. In case ordinary and weighted ordinary linear regression this difference will be zero.
xaxis	Values on the x-axis. One can choose from estimated values of x (xaxis="xhat"), y (xaxis="xhat") or the mean of estimated values of x and y (xaxis="both"). If res.type="optimized" the proper type of residuals for each regression will be drawn.
ref.line	logical value. If ref.line = TRUE (default), the reference line will be drawn.
ref.line.col	reference line color.
ref.line.lty	reference line type.
ref.line.lwd	reference line width.
main	character string specifying the main title of the plot
xlab	label for the x-axis
ylab	label for the y-axis
add.grid	logical value. If add.grid = TRUE (default) the gridlines will be drawn.
...	further graphical parameters

Value

No return value, instead a plot is generated

See Also

getResiduals, plot.mcr, plotDifference, plotBias, compareFit

Examples

```
data(creatinine,package="mcrPioda")
x <- creatinine$serum.crea
y <- creatinine$plasma.crea

# Deming regression fit.
# The confidence intercals for regression coefficients
# are calculated with analytical method
model <- mcreg( x,y,error.ratio=1,method.reg="WDeming", method.ci="jackknife",
                mref.name = "serum.crea", mtest.name = "plasma.crea", na.rm=TRUE )
plotResiduals(model, res.type="optimized", xaxis="both" )
plotResiduals(model, res.type="y", xaxis="yhat")
```

MCResult.printSummary *Print Summary of a Regression Analysis*

Description

Print Summary of a Regression Analysis

Usage

```
MCResult.printSummary(.Object)
```

Arguments

.Object	object of type "MCResult".

Value

No return value

See Also

getCoefficients, getRegmethod

MCResultAnalytical-class *Class "MCResultAnalytical"*

Description

Result of a method comparison based on analytical methods for computing confidence intervals.

Objects from the Class

Object is typically created by a call to function mcreg. Object can be directly constructed by calling newMCResultAnalytical or new("MCResultAnalytical", data, xmean, para, mnames, regmeth, cimeth, error.ratio, alpha, weight).

Slots

xmean: Object of class "numeric" ~~

data: Object of class "data.frame" ~~

para: Object of class "matrix" ~~

mnames: Object of class "character" ~~

regmeth: Object of class "character" ~~

`cimeth`: Object of class "character" ~~

`error.ratio`: Object of class "numeric" ~~

`alpha`: Object of class "numeric" ~~

`weight`: Object of class "numeric" ~~

Extends

Class "MCResult", directly.

Methods

calcResponse `signature(.Object = "MCResultAnalytical")`: ...

printSummary `signature(.Object = "MCResultAnalytical")`: ...

summary `signature(.Object = "MCResultAnalytical")`: ...

Author(s)

Ekaterina Manuilova <ekaterina.manuilova@roche.com>, Andre Schuetzenmeister <andre.schuetzenmeister@roche.c
Fabian Model <fabian.model@roche.com>, Sergej Potapov <sergej.potapov@roche.com>

Examples

```
showClass("MCResultAnalytical")
```

MCResultAnalytical.calcResponse

Calculate Response

Description

Calculate predicted values for given values of the reference-method.

Usage

```
MCResultAnalytical.calcResponse(.Object, x.levels, alpha = 0.05)
```

Arguments

`.Object`	object of class 'MCResultAnalytical'
`x.levels`	numeric vector specifying values of the reference method for which prediction should be made
`alpha`	significance level for confidence intervals

Value

matrix with predicted values with confidence intervals for given values of the reference-method.

`MCResultAnalytical.initialize` *Initialize Method for 'MCResultAnalytical' Objects.*

Description

Initialize Method for 'MCResultAnalytical' Objects.

Usage

```
MCResultAnalytical.initialize(
  .Object,
  data = data.frame(X = NA, Y = NA),
  xmean = 0,
  para = matrix(NA, ncol = 4, nrow = 2),
  mnames = c("unknown", "unknown"),
  regmeth = "unknown",
  cimeth = "analytical",
  error.ratio = 0,
  alpha = 0.05,
  weight = 1
)
```

Arguments

`.Object`	object to be initialized
`data`	empty data.frame
`xmean`	mean value
`para`	empty coefficient matrix
`mnames`	empty method names vector
`regmeth`	string specifying the regression-method
`cimeth`	string specifying the confidence interval method
`error.ratio`	for Deming regression
`alpha`	value specifying the 100(1-alpha)% confidence-level
`weight`	1 for each data point

Value

No return value

MCResultAnalytical.printSummary

Print Regression-Analysis Summary for Objects of class 'MCResultAnalytical'.

Description

Function prints a summary of the regression-analysis for objects of class 'MCResultAnalytical'.

Usage

```
MCResultAnalytical.printSummary(.Object)
```

Arguments

.Object	object of class 'MCResultAnalytical'

Value

No return value

MCResultBCa-class

Class "MCResultBCa"

Description

Result of a method comparison with BCa-bootstrap based confidence intervals.

Objects from the Class

Object is typically created by a call to function mcreg. Object can be directly constructed by calling newMCResultBCa or new("MCResultBCa", data, para, xmean, mnames, regmeth, cimeth, bootcimeth, alpha, glob.coef, glob.sigma, nsamples, nnested, B0jack, B1jack, B0, B1, MX, rng.seed, rng.kind, sigmaB0, sigmaB1, error.ratio, weight,robust.cov).

Slots

glob.sigma: Object of class "numeric" ~~

xmean: Object of class "numeric" ~~

nsamples: Object of class "numeric" ~~

nnested: Object of class "numeric" ~~

B0: Object of class "numeric" ~~

B1: Object of class "numeric" ~~

sigmaB0: Object of class "numeric" ~~

`sigmaB1`: Object of class "numeric" ~~

`MX`: Object of class "numeric" ~~

`bootcimeth`: Object of class "character" ~~

`rng.seed`: Object of class "numeric" ~~

`rng.kind`: Object of class "character" ~~

`glob.coef`: Object of class "numeric" ~~

`B0jack`: Object of class "numeric" ~~

`B1jack`: Object of class "numeric" ~~

`data`: Object of class "data.frame" ~~

`para`: Object of class "matrix" ~~

`mnames`: Object of class "character" ~~

`regmeth`: Object of class "character" ~~

`cimeth`: Object of class "character" ~~

`error.ratio`: Object of class "numeric" ~~

`alpha`: Object of class "numeric" ~~

`weight`: Object of class "numeric" ~~

`robust.cov`: Object of class "character" ~~

Extends

Class "MCResultJackknife", directly. Class "MCResult", by class "MCResultJackknife", distance 2.

Methods

calcResponse `signature(.Object = "MCResultBCa")`: ...

printSummary `signature(.Object = "MCResultBCa")`: ...

summary `signature(.Object = "MCResultBCa")`: ...

Author(s)

Ekaterina Manuilova <ekaterina.manuilova@roche.com>, Andre Schuetzenmeister <andre.schuetzenmeister@roche.c
Fabian Model <fabian.model@roche.com>, Sergej Potapov <sergej.potapov@roche.com>

Examples

```
showClass("MCResultBCa")
```

MCResultBCa.bootstrapSummary

Compute Bootstrap-Summary for 'MCResultBCa' Objects.

Description

Function computes the bootstrap summary for objects of class 'MCResultBCa'.

Usage

```
MCResultBCa.bootstrapSummary(.Object)
```

Arguments

.Object	object of class 'MCResultBCa'

Value

matrix of bootstrap results

MCResultBCa.calcResponse

Calculate Response

Description

Calculate predicted values for given values of the reference-method.

Usage

```
MCResultBCa.calcResponse(
  .Object,
  x.levels,
  alpha = 0.05,
  bootcimeth = .Object@bootcimeth
)
```

Arguments

.Object	object of class 'MCResultBCa'
x.levels	numeric vector specifying values of the reference method for which prediction should be made
alpha	significance level for confidence intervals
bootcimeth	character string specifying the method to be used for bootstrap confidence intervals

Value

matrix with predicted values with confidence intervals for given values of the reference-method.

`MCResultBCa.initialize` *Initialize Method for 'MCResultBCa' Objects.*

Description

Method initializes newly created objects of class 'MCResultBCa'.

Usage

```
MCResultBCa.initialize(
  .Object,
  data = data.frame(X = NA, Y = NA),
  para = matrix(NA, ncol = 4, nrow = 2),
  xmean = 0,
  mnames = c("unknown", "unknown"),
  regmeth = "unknown",
  cimeth = "unknown",
  bootcimeth = "unknown",
  alpha = 0.05,
  glob.coef = c(0, 0),
  glob.sigma = c(0, 0),
  nsamples = 0,
  nnested = 0,
  B0jack = 0,
  B1jack = 0,
  B0 = 0,
  B1 = 0,
  MX = 0,
  rng.seed = as.numeric(NA),
  rng.kind = "unknown",
  sigmaB0 = 0,
  sigmaB1 = 0,
  error.ratio = 0,
  weight = 1,
  robust.cov = "MCD"
)
```

Arguments

`.Object`	object to be initialized
`data`	empty data.frame
`para`	empty coefficient matrix

xmean	0 for init-purpose
mnames	empty method names vector
regmeth	string specifying the regression-method
cimeth	string specifying the confidence interval method
bootcimeth	string specifying the method for bootstrap confidence intervals
alpha	value specifying the 100(1-alpha)% confidence-level
glob.coef	global coefficients
glob.sigma	global sd values for regression parameters
nsamples	number of samples for resampling
nnested	number of inner simulation for nested bootstrap
B0jack	jackknife intercept
B1jack	jackknife slope
B0	intercept
B1	slope
MX	parameter
rng.seed	random number generator seed
rng.kind	type of the random number generator
sigmaB0	SD for intercepts
sigmaB1	SD for slopes
error.ratio	for Deming regression
weight	1 for each data point
robust.cov	"MCD", "SDe" or "Classic" covariance method see rrcov

Value

No return value

`MCResultBCa.plotBootstrapCoefficients`
Plot distribution of bootstrap coefficients

Description

Plot distribution of bootstrap coefficients (slope and intercept).

Usage

```
MCResultBCa.plotBootstrapCoefficients(.Object, breaks = 20, ...)
```

Arguments

.Object	Object of class "MCResultBCa"
breaks	used in function 'hist' (see ?hist)
...	further graphical parameters

Value

No return value

MCResultBCa.plotBootstrapT

Plot distribution of bootstrap pivot T

Description

Plot distribution of bootstrap pivot T for slope and intercept and compare them with t(n-2) distribution.

Usage

```
MCResultBCa.plotBootstrapT(.Object, breaks = 20, ...)
```

Arguments

.Object	Object of class "MCResultBCa".
breaks	Number of breaks in histogram.
...	further graphical parameters

Value

No return value

MCResultBCa.plotBoxEllipses

Plot Box Ellipses of bootstrap coefficients

Description

Plot Box Ellipses of bootstrap coefficients (slope and intercept).

Usage

```
MCResultBCa.plotBoxEllipses(.Object, robust.cov = "MCD")
```

Arguments

.Object	Object of class "MCResultResampling"
robust.cov	Method for covariance. Default "MCD"

Value

No return value

MCResultBCa.printSummary

Print Regression-Analysis Summary for Objects of class 'MCResultBCa'.

Description

Functions prints a summary of the regression-analysis for objects of class 'MCResultBCa'.

Usage

```
MCResultBCa.printSummary(.Object)
```

Arguments

.Object	object of class 'MCResultBCa'

Value

No return value

MCResultJackknife-class

Class "MCResultJackknife"

Description

Result of a method comparison with Jackknife based confidence intervals.

Objects from the Class

Object is typically created by a call to function mcreg. Object can be directly constructed by calling newMCResultJackknife or new("MCResultJackknife", data, para, mnames, regmeth, cimeth, alpha, glob.coef, B0jack, B1jack, error.ratio, weight).

Slots

`glob.coef`: Object of class "numeric" ~~

`B0jack`: Object of class "numeric" ~~

`B1jack`: Object of class "numeric" ~~

`data`: Object of class "data.frame" ~~

`para`: Object of class "matrix" ~~

`mnames`: Object of class "character" ~~

`regmeth`: Object of class "character" ~~

`cimeth`: Object of class "character" ~~

`error.ratio`: Object of class "numeric" ~~

`alpha`: Object of class "numeric" ~~

`weight`: Object of class "numeric" ~~

Extends

Class "MCResult", directly.

Methods

calcResponse `signature(.Object = "MCResultJackknife")`: ...

getRJIF `signature(.Object = "MCResultJackknife")`: ...

plotwithRJIF `signature(.Object = "MCResultJackknife")`: ...

printSummary `signature(.Object = "MCResultJackknife")`: ...

summary `signature(.Object = "MCResultJackknife")`: ...

Author(s)

Ekaterina Manuilova <ekaterina.manuilova@roche.com>, Andre Schuetzenmeister <andre.schuetzenmeister@roche.c
Fabian Model <fabian.model@roche.com>, Sergej Potapov <sergej.potapov@roche.com>

Examples

```
showClass("MCResultJackknife")
```

MCResultJackknife.calcResponse *Calculate Response*

Description

Calculate predicted values for given values of the reference-method.

Usage

```
MCResultJackknife.calcResponse(.Object, x.levels, alpha = 0.05)
```

Arguments

.Object	object of class 'MCResultJackknife'
x.levels	numeric vector specifying values of the reference method for which prediction should be made
alpha	significance level for confidence intervals

Value

matrix with predicted values with confidence intervals for given values of the reference-method.

MCResultJackknife.getJackknifeIntercept *Get-Method for Jackknife-Intercept Value.*

Description

Extracts the intercept value from objects of class 'MCResultJackknife'.

Usage

```
MCResultJackknife.getJackknifeIntercept(.Object)
```

Arguments

.Object	object of class 'MCResultJackknife'

Value

(numeric) jackknife-intercept

`MCResultJackknife.getJackknifeSlope`
Get-Method for Jackknife-Slope Value.

Description

Extracts the slope value from objects of class 'MCResultJackknife'.

Usage

```
MCResultJackknife.getJackknifeSlope(.Object)
```

Arguments

`.Object`	object of class 'MCResultJackknife'

Value

(numeric) jackknife-slope

`MCResultJackknife.getJackknifeStatistics`
Jackknife Statistics

Description

Calculate jackknife mean, bias and standard error.

Usage

```
MCResultJackknife.getJackknifeStatistics(.Object)
```

Arguments

`.Object`	object of class "MCResultJackknife" or "MCResultResampling"

Value

table with jackknife mean, bias and standard error for intercept and slope.

MCResultJackknife.getRJIF

Relative Jackknife Influence Function

Description

Calculate the value of relative jackknife function for each observation.

Usage

```
MCResultJackknife.getRJIF(.Object)
```

Arguments

.Object	object of class "MCResultJackknife" or "MCResultResampling".

Value

a list of the following elements:

slope	numeric vector containing the values of relative jackknife function of slope.
intercept	numeric vector containing the values of relative jackknife function of intercept.

References

Efron, B. (1990) Jackknife-After-Bootstrap Standard Errors and Influence Functions. Technical Report , **N 134**.

MCResultJackknife.initialize

Initialize Method for 'MCResultJackknife' Objects.

Description

Method initializes newly created objects of class 'MCResultAnalytical'.

Usage

```
MCResultJackknife.initialize(
  .Object,
  data = data.frame(X = NA, Y = NA),
  para = matrix(NA, ncol = 4, nrow = 2),
  mnames = c("unknown", "unknown"),
  regmeth = "unknown",
  cimeth = "jackknife",
  alpha = 0.05,
```

```
  glob.coef = c(0, 0),
  B0jack = 0,
  B1jack = 0,
  error.ratio = 0,
  weight = 1
)
```

Arguments

.Object	object to be initialized
data	empty data.frame
para	empty coefficient matrix
mnames	empty method names vector
regmeth	string specifying the regression-method
cimeth	string specifying the confidence interval method
alpha	value specifying the 100(1-alpha)% confidence-level
glob.coef	global coefficients
B0jack	jackknife intercepts
B1jack	jackknife slopes
error.ratio	for Deming regression
weight	1 for each data point

Value

No return value

MCResultJackknife.plotwithRJIF

Plotting the Relative Jackknife Influence Function

Description

The function draws reference method vs. test method as scatter plot. Observations with high influence (relative jackknife influence function is greater than 2) are highlighted as red points.

Usage

```
MCResultJackknife.plotwithRJIF(.Object)
```

Arguments

.Object	object of class "MCResultJackknife" or "MCResultResampling"

Value

No return value

References

Efron, B. (1990) Jackknife-After-Bootstrap Standard Errors and Influence Functions. Technical Report , **N 134**.

Examples

```
#library("mcr")
data(creatinine,package="mcrPioda")
x <- creatinine$serum.crea
y <- creatinine$plasma.crea
# Deming regression fit.
# The confidence intervals for regression coefficients
# are calculated with jackknife method
model <- mcreg( x,y,error.ratio=1,method.reg="Deming", method.ci="jackknife",
                mref.name = "serum.crea", mtest.name = "plasma.crea", na.rm=TRUE )
plotwithRJIF(model)
```

MCResultJackknife.printSummary

Print Regression-Analysis Summary for Objects of class 'MCResult-Jackknife'.

Description

Functions prints a summary of the regression-analysis for objects of class 'MCResultJackknife'.

Usage

```
MCResultJackknife.printSummary(.Object)
```

Arguments

`.Object`	object of class 'MCResultJackknife'

Value

No return value

MCResultResampling-class

Class "MCResultResampling"

Description

Result of a method comparison with resampling based confidence intervals.

Objects from the Class

Object is typically created by a call to function mcreg. Object can be directly constructed by calling newMCResultResampling or new("MCResultResampling", data, para, xmean, mnames, regmeth, cimeth, bootcimeth, alpha, glob.coef, rng.seed, rng.kind, glob.sigma, nsamples, nnested, B0, B1, MX, sigmaB0, sigmaB1, error.ratio, weight,robust.cov).

Slots

glob.coef: Object of class "numeric" ~~

glob.sigma: Object of class "numeric" ~~

xmean: Object of class "numeric" ~~

nsamples: Object of class "numeric" ~~

nnested: Object of class "numeric" ~~

B0: Object of class "numeric" ~~

B1: Object of class "numeric" ~~

sigmaB0: Object of class "numeric" ~~

sigmaB1: Object of class "numeric" ~~

MX: Object of class "numeric" ~~

bootcimeth: Object of class "character" ~~

rng.seed: Object of class "numeric" ~~

rng.kind: Object of class "character" ~~

data: Object of class "data.frame" ~~

para: Object of class "matrix" ~~

mnames: Object of class "character" ~~

regmeth: Object of class "character" ~~

cimeth: Object of class "character" ~~

error.ratio: Object of class "numeric" ~~

alpha: Object of class "numeric" ~~

weight: Object of class "numeric" ~~

robust.cov: Object of class "character" ~~

Extends

Class "MCResult", directly.

Methods

calcResponse signature(.Object = "MCResultResampling"): ...

printSummary signature(.Object = "MCResultResampling"): ...

summary signature(.Object = "MCResultResampling"): ...

Author(s)

Ekaterina Manuilova <ekaterina.manuilova@roche.com>, Andre Schuetzenmeister <andre.schuetzenmeister@roche.c
Fabian Model <fabian.model@roche.com>, Sergej Potapov <sergej.potapov@roche.com>

Examples

```
showClass("MCResultResampling")
```

MCResultResampling.bootstrapSummary

Compute Bootstrap-Summary for 'MCResultResampling' Objects.

Description

Function computes the bootstrap summary for objects of class 'MCResultResampling'.

Usage

```
MCResultResampling.bootstrapSummary(.Object)
```

Arguments

.Object	object of class 'MCResultResampling'

Value

matrix of bootstrap results

MCResultResampling.calcResponse *Calculate Response*

Description

Calculate predicted values for given values of the reference-method.

Usage

```
MCResultResampling.calcResponse(
  .Object,
  x.levels,
  alpha = 0.05,
  bootcimeth = .Object@bootcimeth
)
```

Arguments

.Object	object of class 'MCResultResampling'
x.levels	numeric vector specifying values of the reference method for which prediction should be made
alpha	significance level for confidence intervals
bootcimeth	bootstrap confidence interval method to be used

Value

matrix with predicted values with confidence intervals for given values of the reference-method.

MCResultResampling.initialize *Initialize Method for 'MCResultAnalytical' Objects.*

Description

Method initializes newly created objects of class 'MCResultAnalytical'.

Usage

```
MCResultResampling.initialize(
  .Object,
  data = data.frame(X = NA, Y = NA),
  para = matrix(NA, ncol = 4, nrow = 2),
  xmean = 0,
  mnames = c("unknown", "unknown"),
  regmeth = "unknown",
  cimeth = "unknown",
  bootcimeth = "unknown",
  alpha = 0.05,
  glob.coef = c(0, 0),
  rng.seed = as.numeric(NA),
  rng.kind = "unknown",
  glob.sigma = c(0, 0),
  nsamples = 0,
  nnested = 0,
  B0 = 0,
  B1 = 0,
  MX = 0,
  sigmaB0 = 0,
  sigmaB1 = 0,
  error.ratio = 0,
  weight = 1,
  robust.cov = "MCD"
)
```

Arguments

.Object	object to be initialized
data	empty data.frame
para	empty coefficient matrix
xmean	0 for init-purpose
mnames	empty method names vector
regmeth	string specifying the regression-method
cimeth	string specifying the confidence interval method
bootcimeth	string specifying the method for bootstrap confidence intervals
alpha	value specifying the 100(1-alpha)% confidence-level
glob.coef	global coefficients
rng.seed	random number generator seed
rng.kind	type of the random number generator
glob.sigma	global sd values for regression parameters
nsamples	number of samples for resampling
nnested	number of inner simulation for nested bootstrap

B0	resampling intercepts
B1	resampling slopes
MX	Numeric vector with point estimations of (weighted-)average of reference method values for each bootstrap sample.
sigmaB0	SD for 'B0'
sigmaB1	SD for 'B1'
error.ratio	for Deming regression
weight	1 for each data point
robust.cov	"MCD", "SDe" or "Classic" covariance method see rrcov

Value

No return value

`MCResultResampling.plotBootstrapCoefficients`
Plot distribution of bootstrap coefficients

Description

Plot distribution of bootstrap coefficients (slope and intercept).

Usage

```
MCResultResampling.plotBootstrapCoefficients(.Object, breaks = 20, ...)
```

Arguments

.Object	Object of class "MCResultResampling"
breaks	see function 'hist' (?hist) for details
...	further graphical parameters

Value

No return value

MCResultResampling.plotBootstrapT

Plot distribution of bootstrap pivot T

Description

Plot distribution of bootstrap pivot T for slope and intercept and compare them with t(n-2) distribution.

Usage

```
MCResultResampling.plotBootstrapT(.Object, breaks = 20, ...)
```

Arguments

.Object	Object of class "MCResultResampling".
breaks	Number of breaks in histogram.
...	further graphical parameters

Value

No return value

MCResultResampling.plotBoxEllipses

Plot Box Ellipses of bootstrap coefficients

Description

Plot Box Ellipses of bootstrap coefficients (slope and intercept).

Usage

```
MCResultResampling.plotBoxEllipses(.Object, robust.cov = "MCD")
```

Arguments

.Object	Object of class "MCResultResampling"
robust.cov	Method for covariance. Default "MCD"

Value

No return value

MCResultResampling.printSummary

Print Regression-Analysis Summary for Objects of class 'MCResultResampling'.

Description

Functions prints a summary of the regression-analysis for objects of class 'MCResultResampling'.

Usage

```
MCResultResampling.printSummary(.Object)
```

Arguments

.Object	object of class 'MCResultResampling'

Value

no return value, print the analysis summary

newMCResult

MCResult Object Constructor with Matrix in Wide Format as Input

Description

MCResult Object Constructor with Matrix in Wide Format as Input

Usage

```
newMCResult(
  wdata,
  para,
  sample.names = NULL,
  method.names = NULL,
  regmeth = "Unknown",
  cimeth,
  error.ratio,
  alpha = 0.05,
  weight = rep(1, nrow(wdata))
)
```

Arguments

wdata	measurement data in matrix format. First column reference method (x), second column test method (y).
para	regression parameters in matrix form. Rows: Intercept, Slope. Cols: EST, SE, LCI, UCI.
sample.names	names of individual data points, e.g. barcodes of measured samples.
method.names	names of reference and test method.
regmeth	name of statistical method used for regression.
cimeth	name of statistical method used for computing confidence intervals.
error.ratio	ratio between standard deviation of reference and test method.
alpha	numeric value specifying the 100(1-alpha)% confidence level of confidence intervals (Default is 0.05).
weight	numeric vector specifying the weights used for each point

Value

MCResult object containing regression results.

---

newMCResultAnalytical *MCResultAnalytical object constructor with matrix in wide format as input.*

Description

MCResultAnalytical object constructor with matrix in wide format as input.

Usage

```
newMCResultAnalytical(
  wdata,
  para,
  xmean,
  sample.names = NULL,
  method.names = NULL,
  regmeth = "Unknown",
  cimeth = "analytical",
  error.ratio = error.ratio,
  alpha = 0.05,
  weight = rep(1, nrow(wdata))
)
```

Arguments

wdata	Measurement data in matrix format. First column reference method (x), second column comparator method (y).
para	Regression parameters in matrix form. Rows: Intercept, Slope. Cols: EST, SE, LCI, UCI.
xmean	Global (weighted) mean of x-values.
sample.names	Names of individual data points, e.g. barcodes of measured samples.
method.names	Names of reference and comparator method.
regmeth	Name of statistical method used for regression.
cimeth	Name of statistical method used for computing confidence intervals.
error.ratio	Ratio between standard deviation of reference and comparator method.
alpha	1 - significance level for confidence intervals.
weight	numeric vector specifying the weights used for each point

Value

MCResultAnalytical object containing regression results.

---

newMCResultBCa *MCResultBCa object constructor with matrix in wide format as input.*

Description

MCResultBCa object constructor with matrix in wide format as input.

Usage

```
newMCResultBCa(
  wdata,
  para,
  xmean,
  sample.names = NULL,
  method.names = NULL,
  regmeth = "unknown",
  glob.coef,
  glob.sigma,
  cimeth = "unknown",
  bootcimeth = "unknown",
  nsamples,
  nnested,
  rng.seed,
  rng.kind,
  B0jack,
  B1jack,
```

```
  B0,
  B1,
  MX,
  sigmaB0,
  sigmaB1,
  error.ratio,
  alpha = 0.05,
  weight = rep(1, nrow(wdata))
)
```

Arguments

wdata	Measurement data in matrix format. First column reference method (x), second column comparator method (y).
para	Regression parameters in matrix form. Rows: Intercept, Slope. Cols: EST, SE, LCI, UCI.
xmean	Global (weighted) mean of x-values
sample.names	Names of individual data points, e.g. barcodes of measured samples.
method.names	Names of reference and comparator method.
regmeth	Name of statistical method used for regression.
glob.coef	Numeric vector of length two with global point estimations of intercept and slope.
glob.sigma	Numeric vector of length two with global estimations of standard errors of intercept and slope.
cimeth	Name of statistical method used for computing confidence intervals.
bootcimeth	Bootstrap based confidence interval estimation method.
nsamples	Number of bootstrap samples.
nnested	Number of nested bootstrap samples.
rng.seed	Seed used to call mcreg, NULL if no seed was used
rng.kind	RNG type (string, see set.seed for details) used, only meaningful if rng.seed was specified
B0jack	Numeric vector with point estimations of intercept for jackknife samples.
B1jack	Numeric vector with point estimations of slope for jackknife samples.
B0	Numeric vector with point estimations of intercept for each bootstrap sample.
B1	Numeric vector with point estimations of slope for each bootstrap sample.
MX	Numeric vector with point estimations of (weighted-)average of reference method values for each bootstrap sample.
sigmaB0	Numeric vector with estimation of standard error of intercept for each bootstrap sample.
sigmaB1	Numeric vector with estimation of standard error of slope for each bootstrap sample.
error.ratio	Ratio between standard deviation of reference and comparator method.
alpha	1 - significance level for confidence intervals.
weight	numeric vector specifying the weights used for each point

Value

MCResult object containing regression results.

`newMCResultJackknife` *MCResultJackknife Object Constructor with Matrix in Wide Format as Input*

Description

MCResultJackknife Object Constructor with Matrix in Wide Format as Input

Usage

```
newMCResultJackknife(
  wdata,
  para,
  sample.names = NULL,
  method.names = NULL,
  regmeth = "Unknown",
  glob.coef,
  cimeth = "unknown",
  B0jack,
  B1jack,
  error.ratio = error.ratio,
  alpha = 0.05,
  weight = rep(1, nrow(wdata))
)
```

Arguments

`wdata`	measurement data in matrix format. First column reference method (x), second column test method (y).
`para`	regression parameters in matrix form. Rows: Intercept, Slope. Cols: EST, SE, LCI, UCI.
`sample.names`	names of individual data points, e.g. barcodes of measured samples.
`method.names`	names of reference and test method.
`regmeth`	name of statistical method used for regression.
`glob.coef`	global coefficients
`cimeth`	name of statistical method used for computing confidence intervals.
`B0jack`	jackknife intercepts
`B1jack`	jeckknife slopes
`error.ratio`	ratio between standard deviation of reference and test method.
`alpha`	numeric value specifying the 100(1-`alpha`)% confidence level of confidence intervals (Default is 0.05).
`weight`	numeric vector specifying the weights used for each point

Value

MCResult object containing regression results.

`newMCResultResampling` *MCResultResampling object constructor with matrix in wide format as input.*

Description

MCResultResampling object constructor with matrix in wide format as input.

Usage

```
newMCResultResampling(
  wdata,
  para,
  xmean,
  sample.names = NULL,
  method.names = NULL,
  regmeth = "unknown",
  glob.coef,
  glob.sigma,
  cimeth = "unknown",
  bootcimeth = "unknown",
  nsamples,
  nnested,
  rng.seed,
  rng.kind,
  B0,
  B1,
  MX,
  sigmaB0,
  sigmaB1,
  error.ratio,
  alpha = 0.05,
  weight = rep(1, nrow(wdata))
)
```

Arguments

`wdata`	Measurement data in matrix format. First column reference method (x), second column comparator method (y).
`para`	Regression parameters in matrix form. Rows: Intercept, Slope. Cols: EST, SE, LCI, UCI.
`xmean`	Global (weighted) mean of x-values
`sample.names`	Names of individual data points, e.g. barcodes of measured samples.

method.names	Names of reference and comparator method.
regmeth	Name of statistical method used for regression.
glob.coef	Numeric vector of length two with global point estimations of intercept and slope.
glob.sigma	Numeric vector of length two with global estimations of standard errors of intercept and slope.
cimeth	Name of statistical method used for computing confidence intervals.
bootcimeth	Bootstrap based confidence interval estimation method.
nsamples	Number of bootstrap samples.
nnested	Number of nested bootstrap samples.
rng.seed	Seed used to call mcreg, NULL if no seed was used
rng.kind	RNG type (string, see set.seed for details) used, only meaningful if rng.seed was specified
B0	Numeric vector with point estimations of intercept for each bootstrap sample.
B1	Numeric vector with point estimations of slope for each bootstrap sample.
MX	Numeric vector with point estimations of (weighted-)average of reference method values for each bootstrap sample.
sigmaB0	Numeric vector with estimation of standard error of intercept for each bootstrap sample.
sigmaB1	Numeric vector with estimation of standard error of slope for each bootstrap sample.
error.ratio	Ratio between standard deviation of reference and comparator method.
alpha	1 - significance level for confidence intervals.
weight	numeric vector specifying the weights used for each point

Value

MCResult object containing regression results.

Index

∗ **classes**
MCResult-class, 35
MCResultAnalytical-class, 54
MCResultBCa-class, 57
MCResultJackknife-class, 63
MCResultResampling-class, 70
∗ **datasets**
creatinine, 6
∗ **package**
mcrPioda-package, 4

calcBias, *34*, *39*, *49*
calcBias *(*MCResult.calcBias*)*, 36
calcBias,MCResult-method *(*MCResult-class*)*, 35
calcCUSUM *(*MCResult.calcCUSUM*)*, 37
calcCUSUM,MCResult-method *(*MCResult-class*)*, 35
calcDiff, 5
calcPaBaTiesRatio *(*MCResult.calcPaBaTiesRatio*)*, 38
calcPaBaTiesRatio,MCResult-method *(*MCResult-class*)*, 35
calcResponse, *34*
calcResponse *(*MCResult.calcResponse*)*, 39
calcResponse,MCResult-method *(*MCResult-class*)*, 35
calcResponse,MCResultAnalytical-method *(*MCResultAnalytical-class*)*, 54
calcResponse,MCResultBCa-method *(*MCResultBCa-class*)*, 57
calcResponse,MCResultJackknife-method *(*MCResultJackknife-class*)*, 63
calcResponse,MCResultResampling-method *(*MCResultResampling-class*)*, 70
coef *(*MCResult.getCoefficients*)*, 40
coef,MCResult-method *(*MCResult-class*)*, 35
compareFit, 5, *7*, *34*, *46*, *49*, *52*, *53*
creatinine, 6

getCoefficients, *33*, *54*
getCoefficients *(*MCResult.getCoefficients*)*, 40
getCoefficients,MCResult-method *(*MCResult-class*)*, 35
getData, *33*
getData *(*MCResult.getData*)*, 40
getData,MCResult-method *(*MCResult-class*)*, 35
getErrorRatio *(*MCResult.getErrorRatio*)*, 41
getErrorRatio,MCResult-method *(*MCResult-class*)*, 35
getFitted *(*MCResult.getFitted*)*, 41
getFitted,MCResult-method *(*MCResult-class*)*, 35
getRegmethod, *54*
getRegmethod *(*MCResult.getRegmethod*)*, 42
getRegmethod,MCResult-method *(*MCResult-class*)*, 35
getResiduals, *34*, *41*, *53*
getResiduals *(*MCResult.getResiduals*)*, 42
getResiduals,MCResult-method *(*MCResult-class*)*, 35
getRJIF *(*MCResultJackknife.getRJIF*)*, 67
getRJIF,MCResultJackknife-method *(*MCResultJackknife-class*)*, 63
getWeights *(*MCResult.getWeights*)*, 43
getWeights,MCResult-method *(*MCResult-class*)*, 35

includeLegend, 6, *46*

mc.analytical.ci, 8
mc.bootstrap, 9
mc.calc.bca, 11
mc.calc.quant, 12
mc.calc.quantile, 12

mc.calc.Student, 13
mc.calc.tboot, 14
mc.calcAngleMat, 15
mc.calcAngleMat.R, 15
mc.calcLinnetCI, 16
mc.calcTstar, 16
mc.deming, 17
mc.linreg, 18
mc.make.CIframe, 19
mc.mdemingConstCV, 20
mc.mmdemingConstCV, 21
mc.mmNgdemingConstCV, 22
mc.mmPidemingConstCV, 23
mc.paba, 24
mc.paba.LargeData, 25
mc.PBequi, 27
mc.wdemingConstCV, 28
mc.wlinreg, 29
mcreg, *4*, *10*, *25–27*, 29, *35*, *54*, *57*, *63*, *70*
MCResult, *55*, *58*, *64*, *71*
MCResult-class, 35
MCResult.calcBias, 36
MCResult.calcCUSUM, 37
MCResult.calcPaBaTiesRatio, 38
MCResult.calcResponse, 39
MCResult.getCoefficients, 40
MCResult.getData, 40
MCResult.getErrorRatio, 41
MCResult.getFitted, 41
MCResult.getRegmethod, 42
MCResult.getResiduals, 42
MCResult.getWeights, 43
MCResult.initialize, 43
MCResult.plot, 44
MCResult.plotBias, 47
MCResult.plotDifference, 50
MCResult.plotResiduals, 52
MCResult.printSummary, 54
MCResultAnalytical-class, 54
MCResultAnalytical.calcResponse, 55
MCResultAnalytical.initialize, 56
MCResultAnalytical.printSummary, 57
MCResultBCa-class, 57
MCResultBCa.bootstrapSummary, 59
MCResultBCa.calcResponse, 59
MCResultBCa.initialize, 60
MCResultBCa.plotBootstrapCoefficients, 61
MCResultBCa.plotBootstrapT, 62
MCResultBCa.plotBoxEllipses, 62
MCResultBCa.printSummary, 63
MCResultJackknife, *58*
MCResultJackknife-class, 63
MCResultJackknife.calcResponse, 65
MCResultJackknife.getJackknifeIntercept, 65
MCResultJackknife.getJackknifeSlope, 66
MCResultJackknife.getJackknifeStatistics, 66
MCResultJackknife.getRJIF, 67
MCResultJackknife.initialize, 67
MCResultJackknife.plotwithRJIF, 68
MCResultJackknife.printSummary, 69
MCResultResampling-class, 70
MCResultResampling.bootstrapSummary, 71
MCResultResampling.calcResponse, 72
MCResultResampling.initialize, 72
MCResultResampling.plotBootstrapCoefficients, 74
MCResultResampling.plotBootstrapT, 75
MCResultResampling.plotBoxEllipses, 75
MCResultResampling.printSummary, 76
mcrPioda *(*mcrPioda-package*)*, 4
mcrPioda-package, 4

newMCResult, *35*, 76
newMCResultAnalytical, *54*, 77
newMCResultBCa, *57*, 78
newMCResultJackknife, *63*, 80
newMCResultResampling, *70*, 81

plot *(*MCResult.plot*)*, 44
plot,MCResult-method *(*MCResult-class*)*, 35
plot.mcr, *7*, *34*, *49*, *52, 53*
plotBias, *7*, *34*, *37*, *46*, *52, 53*
plotBias *(*MCResult.plotBias*)*, 47
plotBias,MCResult-method *(*MCResult-class*)*, 35
plotDifference, *7*, *34*, *46*, *49*, *52, 53*
plotDifference *(*MCResult.plotDifference*)*, 50
plotDifference,MCResult-method *(*MCResult-class*)*, 35
plotResiduals, *7*, *34*, *41, 42*, *46*, *49*, *52*

plotResiduals *(*MCResult.plotResiduals*)*,
        52
plotResiduals,MCResult-method
        *(*MCResult-class*)*, 35
plotwithRJIF
        *(*MCResultJackknife.plotwithRJIF*)*,
        68
plotwithRJIF,MCResultJackknife-method
        *(*MCResultJackknife-class*)*, 63
printSummary, *33*
printSummary *(*MCResult.printSummary*)*, 54
printSummary,MCResult-method
        *(*MCResult-class*)*, 35
printSummary,MCResultAnalytical-method
        *(*MCResultAnalytical-class*)*, 54
printSummary,MCResultBCa-method
        *(*MCResultBCa-class*)*, 57
printSummary,MCResultJackknife-method
        *(*MCResultJackknife-class*)*, 63
printSummary,MCResultResampling-method
        *(*MCResultResampling-class*)*, 70

summary *(*MCResult.printSummary*)*, 54
summary,MCResult-method
        *(*MCResult-class*)*, 35
summary,MCResultAnalytical-method
        *(*MCResultAnalytical-class*)*, 54
summary,MCResultBCa-method
        *(*MCResultBCa-class*)*, 57
summary,MCResultJackknife-method
        *(*MCResultJackknife-class*)*, 63
summary,MCResultResampling-method
        *(*MCResultResampling-class*)*, 70