Package 'ModelDataComp'

November 17, 2016

Title Model-Data Comparison

Version 1.0

Date 2016-11-17

Author Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

Maintainer Matthias Forkel <matthias.forkel@geo.tuwien.ac.at>

Description Metrics and plots for model-data comparison. The package provides functions to 1) compute model performance metrics (function ObjFct), to 2) fit empirical models to data (function MultiFit), and to 3) make model-data comparison plots (functions ScatterPlot, plot.ObjFct with barplots and scatterplots of objective functions, TaylorPlot, WollMilchSauPlot to plot means, distributions and correlations between observations and models).

Depends R (>= 3.2.3), yarrr, raster

Imports plyr, fields, lhs, randomForest, rgenoud, foreach, mgcv, plotrix, classInt, quantreg, MASS

License GPL-2

URL

LazyLoad yes

R topics documented:

`ModelDataComp-package`
Model-Data Comparison

Description

Metrics and plots for model-data comparison. The package provides functions to 1) compute model performance metrics, to 2) multiple fit empirical models or regressions, and to 3) make model-data comparison plots.

Details

Package:	ModelDataComp
Type:	Package
Version:	1.0
Date:	2015-11-17
License:	GPL-2

The package provides basic functions to compare different datasets or to compare models with data.

Model performance metrics: The function `ObjFct` computes several model performance metrics such as correlations, absolute and squared error metrics, bias metrics, and modelling efficiency metrics. The computed objective functions metrics can be converted to text strings with `ObjFct2Text` and can be plotted as barplots and scatterplots with `plot.ObjFct`.

Fit empirical models and regressions: The function `MultiFit` allows to fit multiple regression approaches between a response variable and one or several predictor variables. Currently considered regression approaches are linear regression, 2-nd and 3-rd order polynomial regression, smoothing splines, generalized additive models, random forest, and logistic functions. The function `FitLogistic` optimizes parameters of a regression based on additive or multiplicative logistic functions.

Plotting: The package provides several plotting functions to compare model and data. `ScatterPlot` produces a scatter plot (with points, density lines, or point counts as image) and adds a fitting line based on one or several regression approaches as in `MultiFit`. `ScatterPlot` also allows to colour points and to fit regression lines for different groups. The function `plot.ObjFct` is the plotting routine for `ObjFct` objects. It allows to plot barplots and scatterplots of objective function metrics for several subsets of the model-data comparison. The function `TaylorPlot` produces a standard Taylor diagram for model-data comparison and is actually based on the `plotrix` package. The function `WollMilchSauPlot` allows to compare distributions (based on smoothed densities), mean values, and the performance or agreement between datasets and models in a single plot. `WollMilchSauPlot` extends the function `pirateplot` from the `yarrr` package by a

colour gradient that is based on the performance or agreement of models with data. For example, ny model performance metric from `ObjFct` can be plotted as colour gradient in the pirateplot.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

Maintainer: Matthias Forkel <matthias.forkel@geo.tuwien.ac.at>

See Also

greenbrown.r-forge.r-project.org

`AllEqual` *Check if all values in a vector are the same*

Description

This function is used to check if all values in a vector are equal. It can be used for example to check if a time series contains only 0 or NA values.

Usage

```
AllEqual(x)
```

Arguments

`x`	numeric, character vector, or time series of type ts

Value

The function returns TRUE if all values are equal and FALSE if it contains different values.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

Examples

```
# check if all values are equal in the following vectors:
AllEqual(1:10)
AllEqual(rep(0, 10))
AllEqual(letters)
AllEqual(rep(NA, 10))
```

`FitLogistic` *Fit logistic functions*

Description

Fits logistic functions between one or several predictor variables and a response variable. In case of multi-variate fits, logistic functions can be combined either additatively or multiplicatively:
Additive: y = Logistic(x1) + Logistic(x2) + Logistic(xN)
Multiplicative: y = Logistic(x1) * Logistic(x2) * Logistic(xN)

Usage

```
FitLogistic(x, y, additive = FALSE, method = c("genoud"), ninit = 50,
    ...)
```

Arguments

`x`	predictor variables
`y`	response variables
`additive`	Make an additive or multiplicative fit? The default is a multiplicative fit.
`method`	Method to be used for optimization of fit parameters. "genoud" uses genetic optimization (see `genoud`). Can be also one of "Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN", "Brent" (see `optim`). If two methods are provided, the second method is applied on the best result of the first method.
`ninit`	number of inital parameter sets for the optimization. Inital parameter sets will be included in genoud as part of the first generation. Each initial parameter set will be used as starting value within `optim` methods. A higher number of ninit will likely avoid solutions close to local optima but is computationally more expensive.
`...`	further arguments for `genoud` or `optim`

Details

No details.

Value

An object of class 'FitLogistic' which is actually a list.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

References

No reference.

See Also

Logistic

Examples

```
# 1D example
x <- 1:1000
y <- Logistic(c(1, 0.01, 500), x) + rnorm(1000, 0, 0.01)
plot(x, y)
fit <- FitLogistic(x, y)
lines(x, fit$predicted, col="red")
fit$par # fitted parameter

# performance of fit
ScatterPlot(fit$predicted, y, objfct=TRUE)

## 2D example - takes more time
#n <- 1000
#x1 <- runif(n, 0, 100)
#x2 <- runif(n, 0, 100)
#f1 <- Logistic(c(1, 0.1, 50), x1)
#f2 <- Logistic(c(1, -1, 20), x2)
#plot(x1, f1)
#plot(x2, f2)
#y <- f1 * f2 * rnorm(n, 1, 0.1)
#plot(x1, y)
#plot(x2, y)
#fit <- FitLogistic(x=cbind(x1, x2), y)
#ScatterPlot(fit$predicted, y, objfct=TRUE)
```

GetCommonExtent *get the common extent of different rasterLayers*

Description

The function computes the minimum common extent of RasterLayers.

Usage

```
GetCommonExtent(raster.l)
```

Arguments

raster.l	a list of rasterLayers

Value

The function returns an object of class extent

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

Examples

```
#
```

Logistic *Logistic function*

Description

Compute the result of a logistic function and a predictor variable x:
y = mx / (1 + exp(-sl * (x - x0)))

Usage

```
Logistic(par, x, ...)
```

Arguments

`par`	parameters of the logistic function, a vector of length 3 (asymptote mx, slope sl, turning point x0)
`x`	independent variable
`...`	further arguments (not used)

Details

No details.

Value

a vector

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

References

No reference.

See Also

`FitLogistic`

Examples

```
x <- -20:20
par <- c(1, 0.5, 0)
plot(x, Logistic(par, x), type="l")

par <- c(1, 0.2, 0)
plot(x, Logistic(par, x), type="l")

par <- c(10, -1, 0)
plot(x, Logistic(par, x), type="l")

par <- c(-2, -1, 0)
plot(x, Logistic(par, x), type="l")
```

MinMaxNorm *Normalize to a range*

Description

Normalizes values to a given range.

Usage

```
MinMaxNorm(x, ext = c(0, 1), ...)
```

Arguments

x	values
ext	new minimum and maximum values
...	further arguments (unused)

Details

No details.

Value

vector with normalized values

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

References

No reference.

Examples

```
x <- rnorm(10, 0, 2)
n <- MinMaxNorm(x)
plot(x, n)
```

`MultiFit` *Multiple fits*

Description

Fits bivariate or multivariate regressions between a response variable and one or several predictor variables based on multiple fitting methods.

Usage

```
MultiFit(x, y, fits = c("lm", "quantreg", "poly2", "poly3", "spline",
    "gam"), xout = NULL, excl.quantile = c(0, 1), fit.quantile = NULL,
    ...)
```

Arguments

`x`	predictor variables: a vector for bivariate fits, or a matrix or data.frame for multivariate fits
`y`	vector of a response variable
`fits`	One or several fitting methods that should be used, possible options are: lm, quantreg, poly2, poly3, spline, gam, rf, logistic
`xout`	vector or data.frame of predictor variables for which fits should be returned. If NULL, fits are returned along a sequence of x values. This allows the plotting of 2D surfaces in case of two predictor variables (see examples). In case of xout=x, fits are returned for the same x values that were used for fitting.
`excl.quantile`	lower and upper quantiles for which x and y values should be excluded to compute fits. For example, if excl.quantile=c(0, 0.9) all x and y values above the quantile 0.9 will be excluded from fitting.
`fit.quantile`	Perform a fitting to a certain quantile of x? Setting this argument to an value between 0 and 1 allows quantile regression. Therfore `SelectQuantiles` is first used to select along a range of x only the values that are around the specified quantile.
`...`	further arguments (not used)

Details

The following fitting methods are implemented:

- "lm": (multiple) linear regression based on `lm`: lm(y ~ x)
- "quantreg": quantile regression to the median based on `rq`: rq(y ~ x, tau=0.5)
- "poly2": 2nd-order polynomial regression based on `lm`: lm(y ~ poly(x, degree=2))
- "poly3": 3rd-order polynomial regression based on `lm`: lm(y ~ poly(x, degre=3))
- "spline": smoothing spline based on `smooth.spline`: smooth.spline(x, y). This method only works for bivariate fits.
- "gam": generalized additive models using spline smoothing based on `gam`: gam(y ~ s(x))
- "rf": random forest based on `randomForest`: randomForest(y ~ x). This method is not computed by default because it can be computationally expensive.
- "logistic": multiplicative logistic functions based on `FitLogistic`: FitLogistic(x, y). This method is not computed by default because it can be computationally expensive.

Furthermore, ensemble statistics like the mean, median, standard deviation and percentiles are computed from the results of the choosen fitting methods.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

References

No reference.

See Also

`FitLogistic`

Examples

```
# bivariate example
x <- runif(1000, -3, 3) # predictor variable
y <- 0.5 * x + 1 / exp(-0.4 * x) * rnorm(1000, 1, 1) # response variable
plot(x, y, cex=0.5)
fit <- MultiFit(x, y, fits=c("lm", "quantreg", "poly2", "poly3",
   "spline", "gam", "rf", "logistic"))
summary(fit)
cols <- piratepal("basel")
matplot(fit$x, fit[,2:11], type="l", add=TRUE, lty=1, col=cols, lwd=2)
legend("topleft", colnames(fit)[2:11], lty=1, col=cols, lwd=2)

# same example but exclude very high values (> quantile 0.9) from fitting
fit1 <- MultiFit(x, y, excl.quantile=c(0, 0.9))
lines(fit1$x, fit1$ensMean, type="l",lty=1, col="purple", lwd=3)

# to compare fitted with original values compute
```

```
# fits at original predictor variables (xout=x)
fit <- MultiFit(x, y, fits=c("lm", "gam"), xout=x)
of <- ObjFct(fit$lm, y)
of
ScatterPlot(fit$lm, y, objfct=TRUE)
TaylorPlot(fit$lm, y)

# bivariate example with fit to a certain quantile
plot(x, y)
fit <- MultiFit(x, y, fit.quantile=0.9, fits=c("spline", "gam", "poly3", "rf"))
matplot(fit$x, fit[,2:5], type="l", add=TRUE, lty=1, col=cols, lwd=2)
legend("topleft", colnames(fit)[2:5], lty=1, col=cols, lwd=2)

# example with two predictor variables
a <- runif(1000, -3, 3) # 1st predictor variable
b <- runif(1000, 0, 2) # 2nd predictor variable
y <- 1.2 * b + 1 / exp(-0.4 * a) * rnorm(1000, 1, 0.2) # response variable
plot(a, y)
plot(b, y)
fit <- MultiFit(x=data.frame(a, b), y, xout=NULL)
image(x=unique(fit$a), y=unique(fit$b),
   z=matrix(fit$lm, sqrt(nrow(fit))), main="ensMean")

## as 3D plot:
#require(rgl)
#with(data.frame(a, b), plot3d(a, b, y))
#with(fit, surface3d(unique(a), unique(b), ensMean, alpha=0.2, col="red"))

# example with three predictor variables
a <- runif(1000, -3, 3) # 1st predictor variable
b <- runif(1000, 0, 2) # 2nd predictor variable
c <- rnorm(1000, 1, 1) # 3rd predictor variable
y <- 1.2 * b + 1 / exp(-0.4 * a) * c # response variable
x <- data.frame(a, b, c)
fit <- MultiFit(x, y, fits=c("poly2", "rf"), xout=x)
ObjFct(fit$rf, y)
ObjFct(fit$poly2, y)
```

`ObjFct` *Objective functions*

Description

Calculates several model performance metrics.

Usage

```
ObjFct(sim, obs, groups = NULL)
```

Arguments

`sim`	simulated/predicted/modeled values
`obs`	observed/reference values
`groups`	vector of groups to compute objective functions for several subsets

Details

The function computes several model performance metrics. These metrics are commonly used for the evaluation and optimization of environmental models (Janssen and Heuberger 1995, Legates et al. 1999, Krause et al. 2005, Gupta et al. 2009). The following metrics are implemented:

- Pearson correlation coefficient and p-value: `cor.test`
- Spearman correlation coefficient and p-value: `cor.test`
- Slope of linear regression: `lm`
- Coefficient of determination: `lm`
- Metrics as described in Janssen and Heuberger (1995):
  - Average error
  - Normalized average error
  - Fractional mean bias
  - Relative mean bias
  - Fractional variance
  - Variance ratio
  - Kolmogorov-Smirnov statistic: `ks.test`
  - Root mean squared error
  - Normalized RMSE
  - Index of agreement
  - Mean absolute error
  - Normalized mean absolute error
  - Maximal absolute error
  - Median absolute error
  - Upper quartile absolute error
  - Ratio of scatter
  - Modelling efficiency (Nash-Sutcliffe efficiency)
- Percent bias
- Sum squared error
- Mean squared error
- Kling-Gupta efficiency (Gupta et al. 2009):
  - Kling-Gupta efficiency
  - fractional contribution of bias
  - fractional contribution of variance
  - fractional contribution of correlation

Value

An object of class "ObjFct" which is actually a list.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

References

Gupta, H. V., H. Kling, K. K. Yilmaz, and G. F. Martinez (2009), Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling, Journal of Hydrology, 377(1-2), 80-91, doi:10.1016/j.jhydrol.2009.08.003.
Janssen, P. H. M., and P. S. C. Heuberger (1995), Calibration of process-oriented models, Ecological Modelling, (83), 55-66.
Krause, P., D. P. Boyle, and F. Baese (2005), Comparison of different efficiency criteria for hydrological model assessment, Adv. Geosci., 5, 89-97, doi:10.5194/adgeo-5-89-2005.
Legates, D. R., and G. J. McCabe (1999), Evaluating the use of "goodness-of-fit" Measures in hydrologic and hydroclimatic model validation, Water Resour. Res., 35(1), 233-241, doi:10.1029/1998WR900018.

See Also

`plot.ObjFct`, `ObjFct2Text`, `WollMilchSauPlot`

Examples

```
obs <- 1:100 # 'observations'

# simulated and observed values agree
sim <- obs
ObjFct(sim, obs)

# simulation has a bias
sim <- obs - 50
ObjFct(sim, obs)

# negative correlation
sim <- 100:1
ObjFct(sim, obs)

# same mean, same correlation but smaller variance
sim <- 0.5 * obs + 25.25
ObjFct(sim, obs)

# small scatter around observations
sim <- obs * rnorm(100, 1, 0.1)
ObjFct(sim, obs)

# larger scatter around observations
sim <- obs * rnorm(100, 1, 0.8)
ObjFct(sim, obs)
```

```
# bias and larger scatter around observations
sim <- obs * rnorm(100, 2, 0.8)
ObjFct(sim, obs)
ScatterPlot(obs, sim, objfct=TRUE)

# simulation is independent from observations
sim <- rnorm(100, 0, 1)
ObjFct(sim, obs)

# split by groups
sim <- obs * c(rnorm(40, 1, 0.2), rnorm(60, 1.2, 0.5))
groups <- c(rep("subset 1", 40), rep("subset 2", 60))
of <- ObjFct(sim, obs, groups=groups)
of
ScatterPlot(obs, sim, groups=groups, objfct=TRUE)

# convert objective functions to text
ObjFct2Text(of)
ObjFct2Text(of, which="KGE")
ObjFct2Text(of, which=c("R2", "MEF", "KGE"), sep=" ")

# plot scatterplot of two metrics
plot(of)
plot(of, which=c("MEF", "RMSE"))


# analyze relations between objective functions
#----------------------------------------------

# Some objective function metrics are closely related to others.
# This simple example demonstrates the relations bewteen metrics.
# Therefore, several experiments are performed. In each experiment,
# simulation with different biases, correlations, and variance in
# comparison with the observations are created.
# Objective functions are then computed for all experiments.

# the 'observations'
obs <- 1:100

# experiments: create several 'simulations'
n <- 500 # how many experiments?
data <- data.frame(obs=NA, sim=NA, exp=NA)
for (i in 1:n) {
   fbias <- runif(1, 0.01, 2) # factor to create the bais
   fcor <- runif(1, -2.5, 2.5) # factor to create a different correlation
   fsd <- runif(1, 0.1, 2) # factor to create variability
   sim <- fbias * obs^(fcor) # create simulations
   sim <- sim + rnorm(length(sim), 0, fsd*abs(mean(sim, na.rm=TRUE)))
   data <- rbind(data, data.frame(obs=obs, sim=sim, exp=i))
}
data <- na.omit(data)
```

```
# plot the first 5 experiments:
plot(sim ~ obs, data=data[1:500,], col=data$exp[1:500], pch=16)

# compute objective function metrics for each experiment
of <- ObjFct(data$sim, data$obs, groups=data$exp)
hist(of$Cor)

# check relations between metrics
plot(of, c("Cor", "Spearman"))
plot(of, c("Cor", "R2"))
plot(of, c("Cor", "IoA"))
plot(of, c("RMSE", "AE"))
plot(of, c("RMSE", "MEF"))
plot(of, c("KS", "MEF"))
```

`ObjFct2Text` *Convert objective functions to text*

Description

Converts an object of class `ObjFct` to text string.

Usage

```
ObjFct2Text(x, which = c("Cor", "RMSE", "MEF"), sep = ", ", digits = 3)
```

Arguments

`x`	an object of class `ObjFct`
`which`	which objective function metrics should be written as text?
`sep`	separation between metrics
`digits`	digits for rounding

Details

Converts an object of class 'ObjFct' to text string.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

References

No reference.

See Also

`ObjFct`

Examples

```
obs <- rnorm(100) # "observations"
sim <- obs + rnorm(100, 0, 0.8) # simulation
of <- ObjFct(sim, obs)
of

# convert objective functions to text:
ObjFct2Text(of)
ObjFct2Text(of, which="KGE")
ObjFct2Text(of, which=c("R2", "MEF", "KGE"), sep=" ")
```

`plot.ObjFct` *Plot objective functions*

Description

Barplot or scatterplot of objective function metrics. A barplot is created if only one metric is defined in `which`. A scatterplot is created if two metrics are defined.

Usage

```
## S3 method for class 'ObjFct'
plot(x, which = c("Cor", "KS", "AE"), alpha = 0.1, cols = "black",
    pch = NULL, legend = TRUE, xlab = NULL, ylab = NULL, xlim = NULL,
    ylim = NULL, txt = NULL, add = FALSE, ...)
```

Arguments

`x`	an object of class `ObjFct`
`which`	Which objective function metrics should be plotted? If one metric is defined, a barplot will be plotted. If more than one metric is defined, the first metric will be used for the x-axis and the second metric for y-axis of a scatterplot. If a third metric is specified, point sizes in the scatterplot will be scaled.
`alpha`	significance level to plot objetive functions that are not significant different from the optimum with different point symbols. This only works if the the selected objective functions in which have a p-value. Set to NULL to avoid this option.
`cols`	colors for the groups
`pch`	point symbols for scatterplots or for pointsplots (instead of barplot)
`legend`	plot a legend for the significance point symbols?
`xlab`	label for x-axis
`ylab`	label for y-axis
`xlim`	limits for x-axis
`ylim`	limits for y-axis

txt	text for the points
add	add plot to an existing plot? This only works for scatterplots.
...	further arguments to plot

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

References

No reference.

See Also

ObjFct, TaylorPlot, ScatterPlot, WollMilchSauPlot

Examples

```
# create some data
obs <- 1:300
sim <- obs * c(rnorm(100, 1, 0.1), rnorm(100, 1, 0.3), rnorm(100, 1, 0.5))
groups <- c(rep("subset 1", 100), rep("subset 2", 100), rep("subset 3", 100))
ScatterPlot(sim, obs, groups, objfct=TRUE)

of <- ObjFct(sim, obs, groups)
of

# default plot
plot(of)

# barplots with only one metric
plot(of, "Cor")
plot(of, "AE")
plot(of, "MEF")

# options for the scatterplot:
plot(of, c("Cor", "RMSE", "MEF"))
plot(of, c("Spearman", "KS"), alpha=0.05)
plot(of, c("Spearman", "KS"), alpha=NULL)
plot(of, cols=c("black", "red", "blue", "green"))
plot(of, c("Cor", "FV"), pch=0:4)
```

`predict.FitLogistic` *Predict method for logistic function fits*

Description

Predict values based on logistic function fits.

Usage

```
## S3 method for class 'FitLogistic'
predict(object, newdata, ...)
```

Arguments

`object`	an oject as returned by `FitLogistic`
`newdata`	data.frame of predictor variables
`...`	further arguments (not used)

Details

No details.

Value

a vector of predicted values

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

References

No reference.

See Also

`FitLogistic`

Examples

```
x <- 1:1000
y <- Logistic(c(1, 0.01, 500), x) + rnorm(1000, 0, 0.01)
plot(x, y)
fit <- FitLogistic(x, y)
lines(x, fit$predicted, col="red")
```

```
newdata <- data.frame(x=-500:1000)
pred <- predict(fit, newdata)
plot(newdata$x, pred)
```

`print.ObjFct` *Print objective functions*

Description

Prints objective functions

Usage

```
## S3 method for class 'ObjFct'
print(x, ...)
```

Arguments

`x`	an object of class `ObjFct`
`...`	further arguments passed to or from other methods

Details

Prints an object of class `ObjFct`.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

References

No reference.

See Also

`ObjFct`

`RandomLHSPar` *Latin-hypercube sampling of parameters*

Description

RandomLHSPar samples parameters based on random Latin-Hypercube sampling within given parameter ranges

Usage

```
RandomLHSPar(nexp, lower, upper, fixed = rep(FALSE, length(lower)))
```

Arguments

`nexp`	number of samples
`lower`	numeric vector of lower boundary values for each parameter
`upper`	numeric vector of upper borundary values for each parameter
`fixed`	boolean vector: should a parameter kept fixed (TRUE) or should all parameters be sampled (FALSE). If TRUE than the parameter value from the vector lower will be returned.

Value

The function returns a data.frame with sampled parameters.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

Examples

```
lower <- c(0, 100, -50, 1000)
names(lower) <- letters[1:4]
upper <- c(10, 500, 50, 10000)
RandomLHSPar(10, lower, upper)
```

`ScatterPlot` *Enhanced scatterplot with fitting lines*

Description

This function creates a scatterplot with fitting lines. Points and fitting lines can be also computed by groups.

Usage

```
ScatterPlot(x, y, groups = NULL, plot.type = NULL, objfct = FALSE,
    which.objfct = c("Cor", "Pbias", "RMSE"), add = FALSE, density.levels = 50,
    density.labels = FALSE, image.nbrks = 300, fits = c("spline"),
    fit.quantile = NULL, fit.minnobs = 10, fit.groups = TRUE,
    fit.global = TRUE, col.points = "black", col.image = NULL,
    col.density = NULL, col.fit = col.points, col.global = NULL,
    xlab = "", ylab = "", main = "", xlim = NULL, ylim = NULL,
    lwd = 2, lty = 1, quantile.x = c(0.01, 0.99), quantile.y = c(0.01,
        0.99), plot = TRUE, cex = NULL, pch = 21, alpha = NULL,
    ...)
```

Arguments

`x`	vector of x values
`y`	vector of y values
`groups`	vector of grouping variables (optional), in case of NULL all points are treated as one group
`plot.type`	plot type: possible arguments are 'density' to plot density lines, 'points' to plot points, or 'image' to plot point counts as a raster image. In case of NULL, the optimal plot.type will be determined dependent on the number of values. It is also possible to combine plot types (e.g. c("image", "density")).
`objfct`	Compute objective functions and add a 1:1 line to the plot?
`which.objfct`	Which objective functions should be added to legend of the plot if objfct=TRUE?
`add`	add plot to existing plot?
`density.levels`	number of levels for density lines
`density.labels`	add labels to density lines?
`image.nbrks`	minimum number of breaks for the image
`fits`	Fitting methods that should be used. See `MultiFit` for details. If several methods area provided the mean of all methods is plotted.
`fit.quantile`	Compute fit to a certain quantile? If NULL, fits to the mean. Otherwise sa value between 0 and 1 can be specified to fit to a certain quantile.

fit.minnobs	minimum number of observations per group in order to calculate a fitting curve
fit.groups	Plot fit lines for groups?
fit.global	Plot global fit lines?
col.points	color for the points/groups
col.image	color range for image
col.density	color range for density lines
col.fit	colors for fitting lines
col.global	colors for global fittiing line
xlab	a title for the x axis
ylab	a title for the y axis
main	an overall title for the plot
xlim	range for the x axis
ylim	range for the y axis
lwd	line width
lty	line type
quantile.x	lower and upper quantile to exclude extreme x values from plotting and fit calculations
quantile.y	lower and upper quantile to exclude extreme y values from plotting and fit calculations
plot	should a scatterplot be plotted or calculate only fitting lines
cex	size of the points in the plot. If NULL point size will be calculated based on the number of points.
pch	point symbol
alpha	transparency of the points (0...255)
...	further arguments to MultiFit or plot.default

Details

This function plots a scatterplot.

Value

The function returns a list with the computed fitting lines and objective functions per group.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

See Also

ObjFct

Examples

```
# create some data:
n <- 10000
x <- runif(n, 0, 100)
groups <- c(rep(1, 3000), rep(2, 3000), rep(3, 4000))
y <- (3 * x^(1/2) + rnorm(n, 0, x/20)) * groups

# standard plot: not very well distinguishable
plot(x, y)

# ScatterPlot
result <- ScatterPlot(x, y)

# ScatterPlot with coulored groups and fitting lines
result <- ScatterPlot(x, y, groups)

# different plot types
result <- ScatterPlot(x, y, plot.type="points")
result <- ScatterPlot(x, y, plot.type="density")
result <- ScatterPlot(x, y, plot.type=c("image", "density"))

# plot and compute objective functions
result <- ScatterPlot(x, y, groups, objfct=TRUE)
result

# plot fits to upper 0.9 and lower 0.1 quantiles, mean fit from two methods
result <- ScatterPlot(x, y, groups, fits=c("poly3", "spline"),
 fit.quantile=0.9, plot.type="image", fit.global=FALSE)
result <- ScatterPlot(x, y, groups, fits=c("poly3", "spline"),
 fit.quantile=0.1, plot.type=NA, add=TRUE, fit.global=FALSE)
```

SelectQuantiles *Select values around quantiles*

Description

The function selects from a set of points along the range of x values these points that are close to a certain quantile of y. First, x values are classified in n groups. Secondly, the specified quantile of y is computed for each group. Thirdly, all y values in a group that are outside +- 0.05 of the estimated quantile are removed. See examples for an illustration.

Usage

```
SelectQuantiles(x, y, q = 0.5, n = NULL, ...)
```

Arguments

x	vector of x values
y	vector of y values
q	quantile value for which data should be selected
n	number of classes for x
...	further arguments

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

Examples

```
# x and y points
x <- 1:1000
y <- x * rnorm(1000, 1, 3)
plot(x, y)

# select points that are close to the median
q05 <- SelectQuantiles(x, y)
points(q05$x, q05$y, col="red")

# select points that are close to the 0.9 quantile
q09 <- SelectQuantiles(x, y, 0.9)
points(q09$x, q09$y, col="blue")

# select points that are close to the 0.1 quantile
q01 <- SelectQuantiles(x, y, 0.1)
points(q01$x, q01$y, col="purple")

# the selected points can be used for fits to a specific quantile
abline(lm(y ~ x, q09), col="blue")
abline(lm(y ~ x, q01), col="purple")
```

TaylorPlot *Plot a Taylor diagram*

Description

Plot a Taylor diagram. This is a modification of the function `taylor.diagram` in the plotrix package.

Usage

```
TaylorPlot(sim, obs, groups = rep(1, length(sim)), col = NULL,
    plot.combined = FALSE, normalize = TRUE, sd.arcs = TRUE,
    text.groups = NULL, text.obs = "Obs", text.combined = "All",
    pos.cor = TRUE, ...)
```

Arguments

sim	simulated/predicted/modeled values
obs	observed/reference values
groups	numeric vector that split ref and model in different groups, e.g. to indicate different models or subsets of the data
col	color for each group, should have the same length as grouping elements
plot.combined	plot also the combined set, i.e. without splitting by groups?
normalize	whether to normalize the models so that the reference has a standard deviation of 1
sd.arcs	whether to display arcs along the standard deviation axes
text.groups	text labels for the groups
text.obs	text label for the observations
text.combined	text label for the combined model/data set without grouping
pos.cor	whether to display only positive (TRUE) or all values of correlation (FALSE). If NULL, this will depend on the correlations.
...	further arguments as in taylor.diagram

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

References

Taylor, K.E., 2001. Summarizing multiple aspects of model performance in a single diagram. Journal of Geophysical Research 106, 7183-7192.

See Also

ObjFct, plot.ObjFct, WollMilchSauPlot

Examples

```
obs <- rnorm(50, 0, 5)
model1 <- obs + c(rnorm(25, 1, 2), rnorm(25, 4, 0.2))
model2 <- obs + c(rnorm(25, -5, 5), rnorm(25, 4, 0.2))
model3 <- obs + c(rnorm(25, 10, 10), rnorm(25, 4, 0.2))
data <- data.frame(obs=rep(obs, 3), model=c(model1, model2, model3),
   group.models=rep(c("model1", "model2", "model3"), each=50),
   group.models.subsets=rep(c("model1.subset1", "model1.subset2",
   "model2.subset1", "model2.subset2", "model3.subset1", "model3.subset2"), each=25))

TaylorPlot(data$model, data$obs, data$group.models)
TaylorPlot(data$model, data$obs, data$group.models.subsets)
```

WollMilchSauPlot *Compare means, distributions, and correlations of different datasets*

Description

Plots a `pirateplot` with couulors according to model performance. The function can be used to compare means, distributions, and correlations (or any other metric from `ObjFct`) between different datasets and models. Thus this plot is almost an "eier-legende Wollmichsau" (german, animal that produces eggs, wool, milk and meet) for model-data comparison. The plot is based on the function `pirateplot`

Usage

```
WollMilchSauPlot(x, ref = 1, objfct = "Cor", cols = tim.colors(10),
    brks = NULL, names = NULL, main = NULL, xlab = NULL, ylab = NULL,
    xlim = NULL, ylim = NULL, legend = TRUE, legend.only = FALSE,
    point.pch = NA, bar.o = 1, bean.o = 1, inf.o = 0, bean.lwd = 1,
    line.lwd = 1, cut.min = NULL, cut.max = NULL, ...)
```

Arguments

`x`	a data.frame with at least two columns
`ref`	Which column in x is the reference dataset? Also more than one reference can be provided, e.g. ref = c(1,2) will compute the objfct based on the combination of both datasets.
`objfct`	Which objective function metric should be used to create the colour palette? (see `ObjFct`)
`cols`	vector of colors from which the color palette should be interpolated
`brks`	break for colour scale
`names`	names of the datasets

`main`	title of the plot
`xlab`	label for x-axis
`ylab`	label for y-axis
`xlim`	limits for x-axis
`ylim`	limits for y-axis
`legend`	plot a legend?
`legend.only`	plot only a legend?
`point.pch`	point types, default: no points
`bar.o`	opacity of bars
`bean.o`	opacity of beans
`inf.o`	inference bands
`bean.lwd`	line width of beans
`line.lwd`	line width of lines
`cut.min`	Optional minimum value of the beans.
`cut.max`	Optional maximum value of the beans.
`...`	further arguments to `pirateplot`

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

References

No reference.

See Also

`pirateplot`, `ObjFct`, `TaylorPlot`, `WollMilchSauPlot`, `ScatterPlot`

Examples

```
# create some data
obs <- rlnorm(500, 1, 1) # observations
sim1 <- obs * rnorm(500, 1, 0.5) # similar to obs
sim2 <- obs * rnorm(500, 1, 2) # less similar to obs
sim3 <- obs * rnorm(500, 1, 4) # less similar to obs
sim4 <- rlnorm(500, 1, 1) # same distribution but no correlation
sim5 <- rnorm(500, 4.4, 2) # similar mean but different distribution
x <- data.frame(obs, sim1, sim2, sim3, sim4, sim5)
x[x < 0] <- 0
```

```
# default plot
WollMilchSauPlot(x)

# with different objective function as colour
WollMilchSauPlot(x, objfct="IoA")
WollMilchSauPlot(x, objfct="Pbias")
WollMilchSauPlot(x, objfct="FV")

# some other settings, e.g. with plotting points
WollMilchSauPlot(x, ylab="Area (km2)", xlab="Groups", main="Comparison",
 point.pch=16, bar.o=0.5, point.o=0.4)

# different color palettes
WollMilchSauPlot(x, cols=c("blue", "red"))
WollMilchSauPlot(x, cols=rainbow(10))
WollMilchSauPlot(x, cols=heat.colors(5))

# without legend (but using an objective function to colour)
WollMilchSauPlot(x, legend=FALSE)

# only legend
WollMilchSauPlot(x, legend.only=TRUE)

# without using an objective function
WollMilchSauPlot(x, objfct=NULL)

# different example data
obs <- rnorm(500, 5, 1)
sim1 <- obs * rnorm(500, 1, 0.2) # similar to obs
sim2 <- obs * rnorm(500, 2, 1) # bias
sim3 <- obs * rlnorm(500, 1, 0.1) # less similar to obs but highly correlated
x <- data.frame(obs, sim1, sim2, sim3)
WollMilchSauPlot(x)
```