

Package: gamlss.prepdata (via r-universe)

October 7, 2026

Type Package

Title Prepering Data for Fitting a Generalized Additive Model for
Location Scale and Shape

Version 0.1.19

Date 2025-10-08

Description Functions for prepering data to fit a Generalized Additive
Models for Location Scale and Shape from the 'gamlss' or
'gamlss2' package, Stasinopoulos and Rigby (2007)
<[doi:10.18637/jss.v023.i07](https://doi.org/10.18637/jss.v023.i07)>, using for graphical methods
'ggplot2'.

License GPL-2 | GPL-3

URL <https://www.gamlss.com/>

BugReports <https://github.com/gamlss-dev/gamlss.prepdata/issues>

Depends R (>= 4.1.0), gamlss.dist, gamlss (>= 4.3.3), gamlss.foreach

Imports methods, ggridges, ellipse, gamlss.inf, foreach, mgcv,
ggplot2, yaImpute, gamlss.ggplots, acepack

Suggests glmnet, reshape2, igraph, networkD3, grid, gridExtra

LazyLoad yes

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libssl-dev

Repository <https://gamlss-dev.r-universe.dev>

Date/Publication 2026-08-29 09:06:07 UTC

RemoteUrl <https://github.com/gamlss-dev/gamlss.prepdata>

RemoteRef HEAD

RemoteSha ea8727e289a30205c7bd3ba90b023991f3a9ef6b

Contents

gamlss.prepdata-package	2
cor_perm_test	3

data_cor	5
data_dim	7
data_factor	9
data_inter	11
data_leverage	13
data_mcor	14
data_outliers	17
data_part	19
data_rm	21
data_scale	23
data_str	25
data_void	27
data_xyplot	29
Index	33

gamlss.prepdata-package
<i>Prepering Data for Fitting a Generalized Additive Model for Location Scale and Shape</i>

Description

Functions for prepering data to fit a Generalized Additive Models for Location Scale and Shape from the 'gamlss' or 'gamlss2' package, Stasinopoulos and Rigby (2007) <doi:10.18637/jss.v023.i07>, using for graphical methods 'ggplot2'.

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.
The following convention has been used to name the functions:
y_NAME: plots concerning fitted values from a single fitted model
data_NAME: plots concerning residuals from a single fitted model
where NAME refer to different characteristics.

Author(s)

Mikis Stasinopoulos [aut, cre, cph] (ORCID: <<https://orcid.org/0000-0003-2407-5704>>), Robert Rigby [aut] (ORCID: <<https://orcid.org/0000-0003-3853-1707>>), Fernanda De Bastiani [aut] (ORCID: <<https://orcid.org/0000-0001-8532-639X>>), Julian Merder [ctb]
Maintainer: Mikis Stasinopoulos <d.stasinopoulos@gre.ac.uk>

References

- Rigby, R. A. and Stasinopoulos D. M. (2005). Generalized additive models for location, scale and shape,(with discussion), *Appl. Statist.*, **54**, part 3, pp 507-554.
- Rigby, R. A., Stasinopoulos, D. M., Heller, G. Z., and De Bastiani, F. (2019) *Distributions for modeling location, scale, and shape: Using GAMLSS in R*, Chapman and Hall/CRC, doi:10.1201/9780429298547. An older version can be found in <https://www.gamlss.com/>.
- Stasinopoulos D. M. Rigby R.A. (2007) Generalized additive models for location scale and shape (GAMLSS) in R. *Journal of Statistical Software*, Vol. **23**, Issue 7, Dec 2007, doi:10.18637/jss.v023.i07.
- Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC. doi:10.1201/b21973
- Stasinopoulos, M. D., Rigby, R. A., and De Bastiani F., (2018) GAMLSS: a distributional regression approach, *Statistical Modelling*, Vol. **18**, pp, 248-273, SAGE Publications Sage India: New Delhi, India.
- Stasinopoulos, M.D., Kneib, T., Klein, N., Mayr, A. and Heller, G.Z., (2024). *Generalized Additive Models for Location, Scale and Shape: A Distributional Regression Approach, with Applications* (Vol. **56**). Cambridge University Press.
- (see also <https://www.gamlss.com/>).

See Also

[gamlss](#), [gamlss.family](#)

Examples

```
data(rent)
rent |> data_str()
```

cor_perm_test

Permutation and Bootstrap Tests for Pairwise Relationships

Description

There are several functions here;
cor_perm_test performs a permutation test

Usage

```
cor_perm_test(x, y, data = NULL, B = 1000, seed = 123,
  tail = c("one", "two"), fun = cor, ...)

cor_boot(x, y, data = NULL, B = 1000, seed = 123,
  tail = c("one", "two"), fun = cor, ...)
```

Arguments

x	a continuous variable
y	a continuous variable
data	the data where to find x and y
B	The number of simulations or bootstraps
seed	setting the seed number
tail	one or two tail test
fun	the function to calculate the association measure between two continuous variables
...	extra argument pass to the function, fun

Details

Those two functions help to test pairwise associations. The function `cor_perm_test()` uses the Fisher's permutation test while `cor_boot()` using bootstrapping.

Value

The function `cor_perm_test()` returns a R object called `permutationTest` with methods `print` and `plot`.

Author(s)

Mikis Stasinopoulos

References

Rigby, R. A., Stasinopoulos, D. M., Heller, G. Z., and De Bastiani, F. (2019) *Distributions for modeling location, scale, and shape: Using GAMLSS in R*, Chapman and Hall/CRC. doi:10.1201/9780429298547.

Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC. doi:10.1201/b21973

Stasinopoulos, M.D., Kneib, T., Klein, N., Mayr, A. and Heller, G.Z., (2024). *Generalized Additive Models for Location, Scale and Shape: A Distributional Regression Approach, with Applications* (Vol. 56). Cambridge University Press.

(see also <https://www.gamlss.com/>).

See Also

`data_cor`

Examples

```
data_cor(rent99)
pp <- cor_perm_test(rent99$rent, rent99$rent)
pp
plot(pp)
```

data_cor

*Plotting pairwise linear and partial correlations.***Description**

The function `data_cor` takes a data frame and plots the pairwise Pearson's correlation coefficients of all continuous variables in the data.

The function `data_pcor` takes a data frame and plots the pairwise partial Pearson's correlation coefficients of all continuous variables in the data.

The function `data_association` takes a data frame and plots the pairwise association coefficients of all variables in the data. For continuous against continuous variables it shows the absolute value of the Pearson's correlation coefficient, for categorical against categorical it shows Cramer's ϕ , for continuous against categorical it fit an analysis of variance model and reports the square root of the R^2 .

The functions `high_val` and `low_val` take the square matrix generated by the above three functions and shows which pair-wise have a value larger or smaller respectively, than the value specified by the argument by `val`.

Usage

```
data_cor(data, digits = 3, plot = TRUE, diag.off = TRUE,
         lower.tri.off = FALSE, method = c("square", "circle"),
         type = c("pearson", "kendall", "spearman"),
         outline.color = "gray", colors = c("blue", "white", "red"),
         legend.title = "Corr", title, ggtheme = theme_minimal(),
         tl.cex = 12, tl.col = "black", tl.srt = 45, lab = TRUE,
         lab_col = "black", lab_size = 3, circle.size = 20, seed = 123,
         percentage, print.info=TRUE)

data_pcor(data, digits = 3, plot = TRUE, diag.off = TRUE,
         lower.tri.off = FALSE, method = c("square", "circle"),
         outline.color = "gray", colors = c("blue", "white", "red"),
         legend.title = "Corr", title, ggtheme = theme_minimal(),
         tl.cex = 12, tl.col = "black", tl.srt = 45, lab = TRUE,
         lab_col = "black", lab_size = 3, circle.size = 20, seed = 123,
         percentage, print.info=TRUE)

data_association(data, digits = 3, plot = TRUE, diag.off = TRUE,
                lower.tri.off = FALSE, method = c("square", "circle"),
                outline.color = "gray", colors = c("blue", "white", "red"),
                legend.title = "Assoc", title, ggtheme = ggplot2::theme_minimal(),
                tl.cex = 12, tl.col = "black", tl.srt = 45, lab = TRUE,
                lab_col = "black", lab_size = 3, circle.size = 20, seed = 123,
                percentage, print.info=TRUE )

high_val(table, val = 0.90, digits = 3, plot = FALSE, igrph = TRUE)
```

```
low_val(table, val = 0.05, digits = 3, plot = FALSE, igrph = TRUE)
```

Arguments

<code>data</code>	a data frame
<code>table</code>	a correlation table obtained by <code>data_cor(,plot=FALSE)</code> or <code>data_pcor(,plot=FALSE)</code>
<code>digits</code>	the digits for printing the correlation coefficients
<code>plot</code>	whether to plot or not
<code>diag.off</code>	whether to show the diagonal elements
<code>lower.tri.off</code>	whether to show the lower part of the matrix
<code>method</code>	plotting in "square" or "circle"
<code>type</code>	type of correlation c("pearson", "kendall", "spearman")
<code>outline.color</code>	the outline colour
<code>colors</code>	the range of colours
<code>legend.title</code>	title for the legend
<code>title</code>	the main title
<code>ggtheme</code>	the theme for the plot, see package ggthemes for more themes
<code>tl.cex</code>	the text size for the marginal labels
<code>tl.col</code>	the colour of the marginal labels
<code>tl.srt</code>	the angle of the text in the bottom labels of the table
<code>lab</code>	whether to show the correlation coefficients in the table
<code>lab.col</code>	the colour of the lettering of the correlation coefficients
<code>lab.size</code>	the size of the lettering of the correlation coefficients, increase (or decrease) if the default 3 is not appropriate
<code>circle.size</code>	the size of the circles, increase (or decrease) if the default 20 is not appropriate
<code>percentage</code>	this is for big data sets. if more than a million only 10% is plotted, if from 100.00 to a million, 20%, if 50.000 to 100.000, 50% otherwise 100% of the data.
<code>seed</code>	Setting a seed value for selection of the percentage of data (for big data sets)
<code>print.info</code>	whether to print information when cutting the data using <code>data_cut()</code>
<code>val</code>	the threshold value so if the actual value is greater than <code>val</code> it will be reported in <code>high_val()</code>
<code>igrph</code>	if in <code>high_val()</code> , the option <code>plot=TRUE</code> is set, then there are two options for plotting: i) <code>igrph::graph_from_data_frame()</code> with <code>igrph=TRUE</code> or ii) <code>networkD3::simpleNetwork</code> with <code>igrph=FALSE</code>

Value

creates a correlation matrix plot.

Author(s)

Mikis Stasinopoulos

References

Rigby, R. A., Stasinopoulos, D. M., Heller, G. Z., and De Bastiani, F. (2019) *Distributions for modeling location, scale, and shape: Using GAMLSS in R*, Chapman and Hall/CRC. doi:10.1201/9780429298547.

Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC. doi:10.1201/b21973

Stasinopoulos, M.D., Kneib, T., Klein, N., Mayr, A. and Heller, G.Z., (2024). *Generalized Additive Models for Location, Scale and Shape: A Distributional Regression Approach, with Applications* (Vol. 56). Cambridge University Press.

(see also <https://www.gamlss.com/>).

See Also

mcor

Examples

```
data_cor(rent99)
Pearson.cor <- data_cor(rent99, plot=FALSE)

data_pcor(rent99)
partial.cor <- data_pcor(rent99, plot=FALSE)

high_val(partial.cor, val=0.5)
high_val(Pearson.cor, val=0.5)
```

data_dim	<i>Function to get information from data.</i>
----------	---

Description

This is a set of function are designed to help the user to deal with new data sets.

data_dim(): the class, the dimension and the % NA's in the data

data_na_vars(): which variables have NA's and how many

data_na_obs(): which observations have NA's

data_omit(): omit the NA's from the data.

data_names(): The names of the variables in the data.

data_shorter_names(): abbreviate the names up to specified digits.

data_rename() renames some of of the variables.

Usage

```
data_dim(data)

data_na_vars(data)

data_na_obs(data)

data_omit(data)

data_names(data)

data_shorter_names(data, max = 5, newnames)

data_rename(data, oldnames, newnames)
```

Arguments

data	a data frame
max	the maximum number of characters allowed, with default 5. Make sure that you are using enough characters otherwise you could end up with variables with the same name
newnames	New names if not abbreviation is required, as characters
oldnames	the old names as characters

Details

The function `data_dim()` gives the the class, the dimension and the % NA's in the data.

The function `data_na_vars()` gives the number of missing observation for each variable in the data.

The function `data_omit()`: omits the NA's from the data.

The function `data_names()` gives the names of the variables.

The function `data_shorter_names()` takes the current names and abbreviates to max characters.

The function `data_rename()` renames variable from the data.

Value

The function `data_dim()` after printing gives the original data set.

The function `data_na_vars()` prints the number of missing observation for each variable in the data and passes the original data set.

The function `data_omit()`: omits the NA's from the data and passes the new data set.

The function `data_names()` prints the names of the variables in the data and passes the original data

The function `data_shorter_names()` takes the current names and abbreviates to max characters and return the data with shorter names.

Author(s)

Mikis Stasinopoulos, Bob Rigby and Fernanda De Bastiani

References

Rigby, R. A. and Stasinopoulos D. M. (2005). Generalized additive models for location, scale and shape,(with discussion), *Appl. Statist.*, **54**, part 3, pp 507-554.

Rigby, R. A., Stasinopoulos, D. M., Heller, G. Z., and De Bastiani, F. (2019) *Distributions for modeling location, scale, and shape: Using GAMLSS in R*, Chapman and Hall/CRC. An older version can be found in <https://www.gamlss.com/>.

Stasinopoulos D. M. Rigby R.A. (2007) Generalized additive models for location scale and shape (GAMLSS) in R. *Journal of Statistical Software*, Vol. **23**, Issue 7, Dec 2007, <https://www.jstatsoft.org/v23/i07/>.

Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC. Stasinopoulos, M.D., Kneib, T., Klein, N., Mayr, A. and Heller, G.Z., (2024). *Generalized Additive Models for Location, Scale and Shape: A Distributional Regression Approach, with Applications* (Vol. **56**). Cambridge University Press.

(see also <https://www.gamlss.com/>).

See Also

[data_cor](#)

Examples

```
data_dim(rent)
data_na_vars(rent)
data_na_obs(rent)
data_omit(rent)
data_names(rent)
data_shorter_names(rent)
pp=data_rename(rent, c("R", "F1"), c("rent", "floor"))
data_names(pp)
```

data_factor

Changing the Reference Level of Factors

Description

Function to change the reference levels of factors. `y_factor()` takes only one factor `data_factor()` takes a `data.frame`.

Usage

```
y_factor(x, how = c("lower", "higher"))
```

```
data_factor(data, how = c("lower", "higher"))
```

Arguments

data	a data frame
x	a variable
how	which reference level, default the one with fewer observations

Value

A factor or a data.frame depending whether `y_factor()` or `data_factor()` is used.

Note

The idea here is that in model selection maybe we like the first level to be the weakest level so when we select levels the stronger level has a change if different. This is more likely to be useful in `gamlss2()` where the stepwise select levels rather than factors.

Author(s)

Mikis Stasinopoulos

References

Rigby, R. A., Stasinopoulos, D. M., Heller, G. Z., and De Bastiani, F. (2019) *Distributions for modeling location, scale, and shape: Using GAMLSS in R*, Chapman and Hall/CRC. doi:10.1201/9780429298547.

Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC. doi:10.1201/b21973

Stasinopoulos, M.D., Kneib, T., Klein, N., Mayr, A. and Heller, G.Z., (2024). *Generalized Additive Models for Location, Scale and Shape: A Distributional Regression Approach, with Applications* (Vol. 56). Cambridge University Press.

(see also <https://www.gamlss.com/>).

See Also

[data_str](#)

Examples

```
levels(rent$B)
da<-data_factor(rent)
levels(rent$B)
```

data_inter

*Identifying Pair-Wise Interactions in the Data Frames***Description**

The function `data_inter()` is trying to identify pair-wise interactions given the response variable using linear regression methodology. At the moment it works only with continuous response variables.

Usage

```
data_inter(data, response, weights, digits = 3, plot = TRUE,
           lower.tri.off = TRUE,
           method = c("circle", "square"), fit.method = c("linear", "nonlinear"),
           outline.color = "gray", colors = c("blue", "white", "red"),
           legend.title = "Inter", title, ggtheme = theme_minimal(),
           tl.cex = 12, tl.col = "black", tl.srt = 45, lab = TRUE,
           lab_col = "black", lab_size = 3, circle.size = 20,
           percentage, seed = 123, print.info = TRUE)
```

Arguments

<code>data</code>	a data frame
<code>response</code>	the response variable
<code>weights</code>	prior weights
<code>digits</code>	the number of digits in the plot
<code>plot</code>	whether to plot the results
<code>lower.tri.off</code>	whether to show the lower part of the matrix
<code>method</code>	plotting in "square" or "circle"
<code>fit.method</code>	whether in "linear" or "nonlinear"
<code>outline.color</code>	the outline colour
<code>colors</code>	the range of colours
<code>legend.title</code>	title for the legend
<code>title</code>	the main title
<code>ggtheme</code>	the theme for the plot, see package ggthemes for more themes
<code>tl.cex</code>	the text size for the marginal labels
<code>tl.col</code>	the colour of the marginal labels
<code>tl.srt</code>	the angle of the text in the bottom labels of the table
<code>lab</code>	whether to show the correlation coefficients in the table
<code>lab_col</code>	the colour of the lettering of the correlation coefficients
<code>lab_size</code>	the size of the lettering of the correlation coefficients, increase (or decrease) if the default 3 is not appropriate

circle.size	the size of the circles, increase (or decrease) if the default 20 is not appropriate
print.info	whether to print information when cutting the data using data_cut()
percentage	the percentage of data to show if the observation number is too big
seed	Setting a seed value for selection of the percentage of data (for big data sets)

Details

The function `data_inter()` uses the function `z_scores()` to standardize the continuous response variable and then uses linear model fits to establish whether the first order interactions between the x 's are significant or not. It reports the significant level based on Chi-square tests. Note that for large data sets it uses the function `data_cut()` to cut randomly the size of the data in order to use ggplot2 graphs to plot it.

Typically for linear model first order interaction it fits the models $y \sim x_1 + x_2$ and $y \sim x_1 * x_2$, respectively, and calculated significant level based on the difference in deviances. Under the H_0 hypothesis the difference in deviances follow a Chi-square distribution with degrees of freedom based on the difference of the degrees of freedom of the two fitted models.

Value

It produce a plot `plot=TRUE` or a square upper triangular table.

Note

The function `data_inter()` works only for continuous responses.

Author(s)

Mikis Stasinopoulos

References

Rigby, R. A., Stasinopoulos, D. M., Heller, G. Z., and De Bastiani, F. (2019) *Distributions for modeling location, scale, and shape: Using GAMLSS in R*, Chapman and Hall/CRC. doi:10.1201/9780429298547.

Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC. doi:10.1201/b21973

Stasinopoulos, M.D., Kneib, T., Klein, N., Mayr, A. and Heller, G.Z., (2024). *Generalized Additive Models for Location, Scale and Shape: A Distributional Regression Approach, with Applications* (Vol. 56). Cambridge University Press.

(see also <https://www.gamlss.com/>).

See Also

[data_cor](#)

Examples

```
data_inter(rent[, -4, 5], response=R)
```

data_leverage

*Getting the Leverage of All explanatory variables***Description**

The function `data_leverage()` is designed to identify observations in the explanatory variable space with high leverage values and, therefore, potential outliers.

Usage

```
data_leverage(data, response, weights, quantile.value = 0.99, annotate = TRUE,
              line.col = "steelblue4", point.col = "steelblue4",
              annot.col = "darkred", plot = TRUE, title, percentage,
              seed = 123, print.info=TRUE, ...)
```

Arguments

<code>data</code>	The data.frame
<code>response</code>	The response variable (to be excuded)
<code>weights</code>	Prior weights if needed
<code>quantile.value</code>	The quantile values for the identification of high leverage. The default is <code>quantile.value=0.99</code> so only the 1 percent of the high leverage are identifyied as outliers. This should be increase for large data
<code>annotate</code>	whether to annotete the the outliers
<code>line.col</code>	The color of the line
<code>point.col</code>	The colout of the plotting points]
<code>annot.col</code>	The colour of the annotate outliers
<code>plot</code>	whether to show the leverage plot
<code>title</code>	what title to put
<code>percentage</code>	what percentage of data to use in the plot
<code>seed</code>	The seed use to calculate the percentage
<code>print.info</code>	whether to print infomation when cutting the data usinf <code>data_cut()</code>
<code>...</code>	other arguments

Details

The function `data_leverage()` uses the linear model methodology to identify unusual observations as a group within the explnatory variables. It fit a linear model to all explanatory variables in the data, calculate the leverage points and plots them. It identifies one percent of the data as outliers. The line in the plot is calculated as $2(r/n)$ where r is the number of explanatory variables and n the number of obsrvations. $2(r/n)$ is given in the literature as the rule of thumb is that an observation is considered to have high leverage. In practice the value $2(r/n)$ is too low for indetification of outliers. Here we use `quan.val= .99` which identify 1% of the obsrvation with high leverage.

Value

It plots the leverages `plot=TRUE` or identify outliers `plot=FALSE`.

Author(s)

Mikis Stasinopoulos

References

Rigby, R. A. and Stasinopoulos D. M. (2005). Generalized additive models for location, scale and shape,(with discussion), *Appl. Statist.*, **54**, part 3, pp 507-554.

Rigby, R. A., Stasinopoulos, D. M., Heller, G. Z., and De Bastiani, F. (2019) *Distributions for modeling location, scale, and shape: Using GAMLSS in R*, Chapman and Hall/CRC. An older version can be found in <https://www.gamlss.com/>.

Stasinopoulos D. M. Rigby R.A. (2007) Generalized additive models for location scale and shape (GAMLSS) in R. *Journal of Statistical Software*, Vol. **23**, Issue 7, Dec 2007, <https://www.jstatsoft.org/v23/i07/>.

Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC.

Stasinopoulos, M.D., Kneib, T., Klein, N., Mayr, A. and Heller, G.Z., (2024). *Generalized Additive Models for Location, Scale and Shape: A Distributional Regression Approach, with Applications* (Vol. **56**). Cambridge University Press.

(see also <https://www.gamlss.com/>).

See Also

[data_zscores](#),

Examples

```
data_leverage(rent99[, -c(2,9)], response=rent)
rent99[, -c(2,9)] |> data_leverage( response=rent, plot=FALSE)
```

data_mcor

Functions to fit Maximal Correlation to Data

Description

The function `data_mcor()` is using either the function `mcor()` or the function `cor_M()` to find the non-linear maximal correlation between the continuous variabes of the data. Note that the function `cor_M()` is usinf the function `ace()` of the package `acepack` which is is faster for large data.

The function `mcor()` fits a single maximal correlation. It uses the function `'ACE()'`.

The function `ACE()` it takes two variables and produce among other thing the maximal coefficient.

The function `ACE.iter()` it takes two variables and shows the fittings to acheive the maximal coefficient.

Usage

```

data_mcor(data, fun = cor_M, digits = 3, plot = TRUE, diag.off = TRUE,
  lower.tri.off = FALSE, method = c("square", "circle"),
  fit.method = c("P-splines", "loess"),
  outline.color = "gray", colors = c("blue", "white", "red"),
  legend.title = "Corr", title,
  ggtheme = ggplot2::theme_minimal(), tl.cex = 12,
  tl.col = "black", tl.srt = 45, lab = TRUE, lab_col = "black",
  lab_size = 3, circle.size = 20, ...)

mcor(x, y, data = NULL, fit.method = c("loess", "P-splines"),
  nseg = 10, max.df = 6, ...)

ACE(x, y, weights, data = NULL, con_crit = 0.01,
  fit.method = c("loess", "P-splines"),
  nseg = 10, max.df = 6, ...)

ACE.iter(x, y, weights, data = NULL, con_crit = 0.001,
  fit.method = c("loess", "P-splines"),
  nseg = 10, max.df = 6, ...)

```

Arguments

data	a data frame
fun	the function mcor or cor_M
digits	how many digit to show in the correlation table
plot	whether to plot the results
diag.off	whether to show the diagonal entries
lower.tri.off	whether to show the lower triangular prt of the matrix
method	plotting in "square" or "cicle"
fit.method	whether to use "loess" or "P-splines"
outline.color	the outline colour
colors	the range of colours
legend.title	title for the legend
title	the main title
ggtheme	the theme use by ggplot2 with default theme_minimal()
tl.cex	the text size for the marginal labels
tl.col	the colour of the he marginal labels
tl.srt	the angle of the text in the bottom labels of the table
lab	whether to show the correlation coefficients in the table
lab_col	the colour of the lettering of the correlation coefficients
lab_size	the size of the lettering of the correlation coefficients, increase (or decrease) if the defaul 3 is not appropriate

circle.size	the size of the circles, increase (or decrease) if the default 20 is not appropriate
x	a continuous variable
y	a continuous variable
weights	prior weights
con_crit	convergence critirion for "P-splines" fit
nseg	The number of breakpoints n the definition of "P-splines"
max.df	the maximal degree of freedom alowen if "P-splines" is fitted
...	for more arguments

Details

The function `dsata_mcor()` is using the function `mcor()` to find the non-linear maximum correlation between the continuous varaibes of the data.

Value

The function `data_mcor()` produve a table

The function `mcor()` produve the maxomal correlations

The function `ACE()` produve a object classn "ACE", with methods `plot()` ands `print()`

Author(s)

Mikis Stasinopoulos

References

Rigby, R. A., Stasinopoulos, D. M., Heller, G. Z., and De Bastiani, F. (2019) *Distributions for modeling location, scale, and shape: Using GAMLSS in R*, Chapman and Hall/CRC. doi:10.1201/9780429298547.

Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC. doi:10.1201/b21973

Stasinopoulos, M.D., Kneib, T., Klein, N., Mayr, A. and Heller, G.Z., (2024). *Generalized Additive Models for Location, Scale and Shape: A Distributional Regression Approach, with Applications* (Vol. 56). Cambridge University Press.

(see also <https://www.gamlss.com/>).

See Also

`data_cor`

Examples

```
data_mcor(rent99)
max.cor <- data_mcor(rent99, plot=FALSE)

high_val(max.cor, val=0.5)
```

Description

Those functions identify outliers in variables in the data.

The function `data_outliers()` takes a `data.frame` and applies the `y_outliers()` function on all its continuous variables.

The function `y_outliers()` takes a continuous (numeric) variable, power transform it (if it takes only positive values). Finally either calculates the z-scores (after fitting a 'SHASHo' distribution) and identify observations with values of z-score higher than the option value (default 3) or used the IQR method which identify points value $\times$ IQR way from the median. The option is "zscores" or "quantile", respectively.

The function `y_outliers_both()` takes one continuous variable and identify outliers using both the "zscores" and the "quantile" method. It prints or the union or the intersect of the two resulting observations indices.

The function `y_outliers_by()` takes one continuous variable and a factor and identify outliers within each level of the factor.

The function `y_outliers_loop()` takes one continuous variable and identify outliers by going through a loop first by weighing out observations and then refitting the model.

The function `y_outliers_z()` takes one continuous variable and possibly a factor and identify outliers or by a single fit or by repeated fits where outliers observation from previous fits are removed (equivalent to `y_outliers_loop()`)

Usage

```
data_outliers(data, value, min.distinct = 50, family = SHASHo,
              type = c("zscores", "quantile") )

y_outliers(x, value, family = SHASHo, type = c("zscores", "quantile"),
           transform = TRUE)

y_outliers_both(x, value, family = SHASHo,
               method = c("intersect", "union"))

y_outliers_by(x, by, family = SHASHo, type = c("zscores", "quantile"))

y_outliers_z(x, by, value, family = SHASHo, transform = TRUE, loop = FALSE)

y_outliers_loop(x, value, family = SHASHo, type = c("zscores", "quantile"),
               transform = TRUE)
```

Arguments

data	a data frame
x	a continuous variable
by	a factor partitioning the data
loop	whether to loop for identify more outliers
transform	whether to transform using a power transformation
value	max value from which the absolute value of the z-scores should be greater to identify outliers
min.distinct	if a variable has less distinct values than min.distinct is excluded
family	the distribution family used for standardization
type	for y_outliers() and data_outliers() the type of outlier detection "zscores" or "quantile"
method	for y_outliers() and data_outliers() the type of outlier detection "zscores" or "quantile"
for_y_outliers_both()	whether the "intersect" (default) or the "union" of the two sets will be used

Details

the continuous variables are power transformed and then standardized

Value

return a list

Author(s)

Mikis Stasinopoulos

References

- Rigby, R. A. and Stasinopoulos D. M. (2005). Generalized additive models for location, scale and shape,(with discussion), *Appl. Statist.*, **54**, part 3, pp 507-554.
- Rigby, R. A., Stasinopoulos, D. M., Heller, G. Z., and De Bastiani, F. (2019) *Distributions for modeling location, scale, and shape: Using GAMLSS in R*, Chapman and Hall/CRC. An older version can be found in <https://www.gamlss.com/>.
- Stasinopoulos D. M. Rigby R.A. (2007) Generalized additive models for location scale and shape (GAMLSS) in R. *Journal of Statistical Software*, Vol. **23**, Issue 7, Dec 2007, <https://www.jstatsoft.org/v23/i07/>.
- Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC.
- Stasinopoulos, M.D., Kneib, T., Klein, N., Mayr, A. and Heller, G.Z., (2024). *Generalized Additive Models for Location, Scale and Shape: A Distributional Regression Approach, with Applications* (Vol. **56**). Cambridge University Press.
- (see also <https://www.gamlss.com/>).

See Also[data_names](#)**Examples**

```
da <- rent99[,-2]
data_outliers(da)
```

data_part

*A function to partition a data frame***Description**

The function `data_part()` takes a `data.frame` and creates a new identical `data.frame` with an extra factor called `partition` which can be used to allocate data in different data sets.

- i) if the option of the function is `partition=2L` the factor has two levels `train`, and `test`.
- ii) if the option of the function is `partition=3L` the factor has three levels `train`, for training data, `val` for validation data and `test` for test data.
- iii) if the option of the function is `partition > 4L` say `K` then the levels of the factor are "1", "2"...`"K"`. The factor then can be used to identify `K`-fold cross validation sets (up to `K=20`).

The function `data_part_list()` in parform a similar task like the function `data_part()` but instead of adding a factor to the data creates a list with elements `data.frames`. Note that this function do allow `K`-fold cross-validation creation (up to 20 `K`-folds), see also the function `data_Kfold()`.

The function `data_boot_index()` takes a `data.frame` and produces two list of length `K`, the in-bag list, `IB`, and the out-of-bag list, `OOB`.

The function `data_boot_weights()` takes a `data.frame` and produces a matrix of dimensions `n x K` which column of wich can be used as prior weight in a regression situation.

The function `data_Kfold()` takes a `data.frame` and produces a matrix of indeces which then can be used to fit diffetent sections of the data for cross validation.

The function `data_cut()` takes a `data.frame` and selects randomly specified % of the data. It is usually applied to graphical function =to reduce time for plotting; For `data.frames` with more than 50.000 observations is automatically select part of the data.

Usage

```
data_part(data, partition = 2L, probs, setseed = 123, ...)
```

```
data_part_list(data, partition = 2L, probs, setseed = 123, ...)
```

```
data_boot_index(data, B = 10, setseed = 123)
```

```
data_boot_weights()(data, B = 10, setseed = 123)
```

```
data_Kfold_index()(data, K = 10, setseed = 123)

data_Kfold_weights()(data, K = 10, setseed = 123)

data_cut(data, percentage, seed = 123, print.info = TRUE)
```

Arguments

data	a <code>data.frame</code>
partition	2, 3 or a number less than 20
K	the number of partitions, (maximum 20 for CV)
B	the number of bootstrap samples
probs	probabilities for the random selection
setseed	setting the seed so the process can be repeated
percentage	The percentage of data to keep. If set, i.e. <code>percentage=0.5</code> only 50% are kept otherwise for large data set (more than 50.000) only percentage of data are kept.
seed	the <code>set.seed()</code> argument
print.info	whether to print information when cutting the data using <code>data_cut()</code>
...	extra arguments

Value

The functions `data_part()`, `data_part_list()`, `data_boot_index()`, `data_boot_weights()`, `data_Kfold()` produce a `data.frame`, lists or matrices to be later used for data partition during the fitting process. The function `data_cut()` randomly select part of the data.

Author(s)

Mikis Stasinopoulos, Bob Rigby and Fernanda De Bastiani

References

- Rigby, R. A., Stasinopoulos, D. M., Heller, G. Z., and De Bastiani, F. (2019) *Distributions for modeling location, scale, and shape: Using GAMLSS in R*, Chapman and Hall/CRC. An older version can be found in <https://www.gamlss.com/>.
- Stasinopoulos D. M. Rigby R.A. (2007) Generalized additive models for location scale and shape (GAMLSS) in R. *Journal of Statistical Software*, Vol. **23**, Issue 7, Dec 2007, <https://www.jstatsoft.org/v23/i07/>.
- Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC.
- Stasinopoulos, M.D., Kneib, T., Klein, N., Mayr, A. and Heller, G.Z., (2024). *Generalized Additive Models for Location, Scale and Shape: A Distributional Regression Approach, with Applications* (Vol. **56**). Cambridge University Press.
- (see also <https://www.gamlss.com/>).

See Also[data_str](#)**Examples**

```

da <- data_part(rent)
head(da)
mosaicplot(table(da$partition))

da.train <- subset(da, partition=="train")
da.test <- subset(da, partition=="test")
dim(da.train)
dim(da.test)

allda <- data_part_list(rent)
dim(allda[[1]]) # training data
dim(allda[[2]]) # test data

```

data_rm

*Functions operating on variables in the data***Description**

There are several function operating on a `data.frame` and export a `data.frame`. The functions are

- 1) `data_rm()`: this function removes the variables specified by `vars` from the `data.frame`. Note that `vars` can take either character names or numbers.
- 2) `data_rm1val()`: This function looks for variables with a unique distinct value (most likely factors left from a previous `subset()` operation) and remove them from the data.
- 3) `data_exclude_class()`: This function looks for variable (columns) of a specified 'R' class and remove them from the data. The default class is "factor".
- 4) `data_only_continuous()`: This function pick up only the continuous variable in the `data.frame`.
- 5) `data_select()`: This function select only the variables in the `vars` list and save the data.

Usage

```

data_rm(data, vars)

data_rm1val(data)

data_exclude_class(data, class.out = "factor")

data_only_continuous(data)

data_select(data, vars)

data_rmNAvars(data)

```

Arguments

data	a data frame
vars	selected variables (columns from the data frame)
class.out	a specific variable class to be excluded from the data frame

Details

All the above functions can be used for piping i.e. `da |> data_rm1val()`.

Value

return a `data.frame`

Author(s)

Mikis Stasinopoulos, Bob Rigby and Fernanda De Bastiani

References

Rigby, R. A., Stasinopoulos, D. M., Heller, G. Z., and De Bastiani, F. (2019) *Distributions for modeling location, scale, and shape: Using GAMLSS in R*, Chapman and Hall/CRC. An older version can be found in <https://www.gamlss.com/>.

Stasinopoulos D. M. Rigby R.A. (2007) Generalized additive models for location scale and shape (GAMLSS) in R. *Journal of Statistical Software*, Vol. **23**, Issue 7, Dec 2007, <https://www.jstatsoft.org/v23/i07/>.

Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC.

Stasinopoulos, M.D., Kneib, T., Klein, N., Mayr, A. and Heller, G.Z., (2024). *Generalized Additive Models for Location, Scale and Shape: A Distributional Regression Approach, with Applications* (Vol. **56**). Cambridge University Press.

(see also <https://www.gamlss.com/>).

See Also

`data_cor`

Examples

```
library(gamlss)
da <- rent |> data_rm( vars=c("Sp", "Sm"))
head(da)

da<- rent |> data_exclude_class()
head(da)

da<- data_only_continuous(rent)
head(da)
```

```
da <- rent |> data_select( vars=c("R", "F1", "A"))
head(da)
```

data_scale

Scaling Continuous Variables in Data

Description

The function `data_scale()` takes a `data.frame` and creates a new data set with all continuous variable standardised. The standardization can be

- i) mean 0 variance 1 which is equivalent to have the options `scale.to="z-scores"` and `family="NO"`. That is the z-scores (residuals) after fitting a normal distribution to a continuous variable
- ii) A more general z-score using say `scale.to="z-scores"` and `family="SHASH"` in which case correction to the skewness and kurtosis is done to the specified variable. Finally,
- iii) the range is restricted from zero to one, i.e. `scale.to="0to1"`.

The function `data_vars2data()` creates a new dataframe where the continuous variables are remain the same or become polynomials and the factors becomes dummy variables.

The function `data_formulae()` takes a `data.frame` and creates four formulae

- 1) The first contains the response and all main effects of the variables in the data i.e. $R \sim F1 + A + K + loc$
- 2) The second contains the response and all first order interactions i.e $R \sim (F1 + A + K + loc)^2$
- 3) The third contains all main effects of the variables in the data with no response i.e. $\sim F1 + A + K + loc$
- 2) The fourth contains all first order interactions with no response i.e $\sim (F1 + A + K + loc)^2$

The function `data_form2X` takes a `data.frame` and a formula and creates a design matrix.

Usage

```
data_scale(data, response, position.response = NULL,
           scale.to = c("z-scores", "0to1"), family = "NO",
           scale.response = FALSE)

data_vars2data(data, response, exclude = NULL,
               type = c("main.effect", "first.order"),
               weights = NULL, nonlinear = FALSE, basis = "poly",
               arg = 2)

data_formulae(data, response)

data_form2X(data, formula, response,
            scale.to = c("no", "z-scores", "0to1"),
            family = NO)
```

Arguments

data	A data frame
response	The name of the response variable
position.response	or the position of the response variable in the data.
scale.to	how to scale by normalization, scale.to="z-scores" or range 0 to 1 scale.to="0to1".
family	The family used in the standarization, defaul is family="NO" but any other continuous family from $-Inf$ to $+Inf$ will do i.e. family="SHASH".
scale.response	whether to scale also the response. The default value is scale.response=FALSE because in GAMLSS we are hoping to get the right family for the response.
exclude	which variable to exclude
type	whether the main effects only ot dummies for first order interactions also
weights	Prior weights when the matrix is created
nonlinear	Im not sure what it does?
basis	what basis should be used for non linearities, deault is poly()
arg	the argument for the basis i.e 3
formula	the formula to create the design matrix

Value

A data frame is return with all continous variables standarised.

Author(s)

Mikis Stasinopolos

References

- Rigby, R. A. and Stasinopoulos D. M. (2005). Generalized additive models for location, scale and shape,(with discussion), *Appl. Statist.*, **54**, part 3, pp 507-554.
- Rigby, R. A., Stasinopoulos, D. M., Heller, G. Z., and De Bastiani, F. (2019) *Distributions for modeling location, scale, and shape: Using GAMLSS in R*, Chapman and Hall/CRC. An older version can be found in <https://www.gamlss.com/>.
- Stasinopoulos D. M. Rigby R.A. (2007) Generalized additive models for location scale and shape (GAMLSS) in R. *Journal of Statistical Software*, Vol. **23**, Issue 7, Dec 2007, <https://www.jstatsoft.org/v23/i07/>.
- Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC.
- Stasinopoulos, M.D., Kneib, T., Klein, N., Mayr, A. and Heller, G.Z., (2024). *Generalized Additive Models for Location, Scale and Shape: A Distributional Regression Approach, with Applications* (Vol. **56**). Cambridge University Press.
- (see also <https://www.gamlss.com/>).

See Also[data_zscores](#)**Examples**

```

rent[, -c(4,5)] |> data_scale(, response=R)|> head()

rent[, -c(4,5)] |> data_vars2data( response=R, nonlinear = TRUE,
                                basis = "poly", arg=3) -> D
D |> head()
D |> str()

ff <- data_formulae(D, response=R)
ff

head(data_form2X(D, formula=ff[[1]], response=R))

```

data_str

*Function applied to data***Description**

his is a set of function are designed to help the user to deal with the structure of new data sets.

Usage

```

data_str(data, min.values = 100, min.levels = 10)

y_distinct(var)

data_distinct(data, get.distinct = FALSE, print=TRUE)

data_cha2fac(data, show.str = FALSE)

data_few2fac(data, max.levels = 5, show.str = FALSE)

data_int2num(data, min.values = 50, show.str = FALSE)

data_fac2num(data, vars)

```

Arguments

data	a data frame
min.values	the minimal value distinct values before warning
max.levels	the maximum value for distinct values in the variable
min.levels	the minimal value distinct levels before warning
var	a vector
vars	a character vector with names from the data
get.distinct	TRUE if you need to save the values FALSE if not
show.str	whether to show the structure
print	TRUE or FALSE

Details

The function `data_str()` gives the structure of the data set.

The function `data_distinct()` gives the distinct values of the vectors in the data set

The function `y_distinct()` gives the distinct values of single vector

The function `data_cha2fac()` transforms all character vectors to factors

The function `data_few2fac()` transform all vectors with fewer values than `min.levels` into factors

The function `data_int2num()` transform all integer vectors with more values than `min.values` into numeric

The function `data_fac2num()` transform selected variables factors into numeric vectors

Author(s)

Mikis Stasinopoulos, Bob Rigby and Fernanda De Bastiani

References

Rigby, R. A. and Stasinopoulos D. M. (2005). Generalized additive models for location, scale and shape,(with discussion), *Appl. Statist.*, **54**, part 3, pp 507-554.

Rigby, R. A., Stasinopoulos, D. M., Heller, G. Z., and De Bastiani, F. (2019) *Distributions for modeling location, scale, and shape: Using GAMLSS in R*, Chapman and Hall/CRC. An older version can be found in <https://www.gamlss.com/>.

Stasinopoulos D. M. Rigby R.A. (2007) Generalized additive models for location scale and shape (GAMLSS) in R. *Journal of Statistical Software*, Vol. **23**, Issue 7, Dec 2007, <https://www.jstatsoft.org/v23/i07/>.

Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC.

Stasinopoulos, M.D., Kneib, T., Klein, N., Mayr, A. and Heller, G.Z., (2024). *Generalized Additive Models for Location, Scale and Shape: A Distributional Regression Approach, with Applications* (Vol. **56**). Cambridge University Press.

(see also <https://www.gamlss.com/>).

See Also[data_dim](#)**Examples**

```

data_str(rent)
data_distinct(rent)
data_cha2fac(rent)
data_few2fac(rent)
data_int2num(rent)

```

data_void

*Finding the Percentage of Empty Spaces***Description**

The function void() is looking for the % of empty spaces in the direction of two variables x and y.

The function data_void() is looking pair-wise for empty spaces in all the continuous variables in the data set.

Usage

```

data_void(data, digits = 3, plot = TRUE, diag.off = TRUE,
          lower.tri.off = FALSE,
          method = c("square", "circle"),
          outline.color = "gray", colors = c("blue", "white", "red"),
          legend.title = "Void", title, ggtheme = ggplot2::theme_minimal(),
          tl.cex = 12, tl.col = "black", tl.srt = 45, lab = TRUE,
          lab.col = "black", lab.size = 3, circle.size = 20, seed = 123,
          percentage, print.info = TRUE)

```

```
void(x, y, plot = TRUE, print = TRUE, table.length)
```

Arguments

data	A data frame
digits	the digits for printing the correlation coefficients
plot	whether to plot or not
diag.off	whether to show the diagonal elements
lower.tri.off	whether to show the lower part of the matrix
method	plotting in "square" or "circle"
outline.color	the outline colour
colors	the range of colours

legend.title	title for the legend
title	the main title
ggtheme	the theme for the plot, see package ggthemes for more themes
tl.cex	the text size for the marginal labels
tl.col	the colour of the marginal labels
tl.srt	the angle of the text in the bottom labels of the table
lab	whether to show the correlation coefficients in the table
lab.col	the colour of the lettering of the correlation coefficients
lab.size	the size of the lettering of the correlation coefficients, increase (or decrease) if the default 3 is not appropriate
circle.size	the size of the circles, increase (or decrease) if the default 20 is not appropriate
seed	the <code>set.seed()</code> value
percentage	the percentage of data to show if the observation number is too big
print.info	whether to print information when cutting the data using <code>data_cut()</code>
x	the first variable in <code>void()</code>
y	the second variable in <code>void()</code>
print	whether to print the results
table.length	the table length (if signing is calculated automatically)

Details

The functions `void()` and `data_void()` work with discretising the data in the x and y direction and then calculate the % of zeros. By discretising the data we mean cut both variable x variables and y, at an equal spaced grid of k points and create a (k x k) dimensional matrix containing the number of data points in the grid. The problem thought, with any attempt to calculate the % of empty spaces is that by increasing k in the x and y directions would result more zeros cells and therefore more % empty spaces. To avoid this we need a way to stop the discretization at a stage before the data become too sparse. The way this is done in the current function is the following;

i) If the n points (x,y) are randomly allocated we would expect the number of counts in the cells of the matrix of a discretised two dimensional data set to be Poisson distributed with a probability for zeros equal to $\exp(-\mu)$ where μ is the mean of the Poisson distribution. That is, under the null hypothesis that the n points are spread randomly we expect some of the cell to be zero with probability $\exp(-\mu)$. Given n the number of observations, we can use this information to find out at which discretation point k we should stop.

ii) To identify at which stage k we should stop for given number of observations say n, we have generated randomly from a uniform distribution n values for x and y. We use those values to calculate at which point k this will give a probability of zero close to 0.05. We calculate those probabilities using $\exp(-xbar)$ where xbar is the mean of the cells. By doing this we found that the following is holding; $\log k = -0.516 + 0.498 \log n$. This equation provide us with an easy way to calculate k given n.

Value

It produce a value between zero and 1.

Author(s)

Mikis Stasinopoulos

References

Rigby, R. A., Stasinopoulos, D. M., Heller, G. Z., and De Bastiani, F. (2019) *Distributions for modeling location, scale, and shape: Using GAMLSS in R*, Chapman and Hall/CRC. doi:10.1201/9780429298547.

Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC. doi:10.1201/b21973

Stasinopoulos, M.D., Kneib, T., Klein, N., Mayr, A. and Heller, G.Z., (2024). *Generalized Additive Models for Location, Scale and Shape: A Distributional Regression Approach, with Applications* (Vol. 56). Cambridge University Press.

(see also <https://www.gamlss.com/>).

See Also

[data_cor](#)

Examples

```
void(rent$A, rent$F1)
data_void(rent)
```

data_xyplot

Plotting the response against the explanatory variables

Description

The function `data_xyplot()` plots the response against all other variables in a given data set.

The function `data_plot()` plots all variables individually.

The function `data_bucket()` plots the bucket plot for all continuous variables.

The function `data_zscores()` calculates and plots the z-scores (obtained after fitting the SHASHo distribution) for all continuous variables.

The function `y_zscores()` calculates and plots the z-scores (obtained after fitting the SHASHo distribution) for a single variable.

The function `data_response()` calculates and plots the z-scores (obtained after fitting the SHASHo distribution) for the response variable.

Usage

```

data_xyplot(data, response, point.size = 0.5, nrow = NULL,
             print.info = TRUE,
             ncol = NULL, percentage, seed = 123, max.levels = 10,
             plots.per.page = 9, one.by.one = FALSE, title, text.x.angle = 0, ...)

data_plot(data, value = 3, hist.col = "black", hist.fill = "white",
           dens.fill = "#FF6666", nrow = NULL, ncol = NULL,
           percentage, seed = 123, print.info = TRUE,
           plot.hist = TRUE, plots.per.page = 9,
           one.by.one = FALSE,
           title, ...)

data_bucket(data, value = 3, max.levels = 20,
            nrow = NULL, ncol = NULL, plots.per.page = 9,
            one.by.one = FALSE, title, percentage, seed = 123,
            print.info = TRUE,
            ...)

y_zscores(x, weights, family = SHASHo, value = 3, plot = TRUE,
           hist = FALSE, transform = FALSE, ...)

data_zscores(data, plot = TRUE, hist=FALSE, value = 3, family = SHASHo,
             max.levels = 10, hist.col = "black", hist.fill = "white",
             dens.fill = "#FF6666", nrow = NULL, ncol = NULL,
             plots.per.page = 9, one.by.one = FALSE, title, print.info = TRUE,
             percentage, seed = 123,...)

data_response(data, response, plot = TRUE,
              percentage, seed = 123, print.info = TRUE)

```

Arguments

<code>data</code>	a data frame
<code>x</code>	a single variable
<code>weights</code>	prior weights
<code>transform</code>	whether to use a power transformation or not
<code>family</code>	a gamlss distribution family (continuous)
<code>response</code>	the response variable should be in the data
<code>point.size</code>	the size of points in scatter plots
<code>nrow</code>	the number of rows in the plot
<code>ncol</code>	the number of columns in the plot
<code>plots.per.page</code>	maximum plots per page
<code>one.by.one</code>	whether plotted individually

value	value to identify outliers if y_dots is used i.e. for upper tail an outliers is if it is greater than $Q_3 + \text{value} * IQ$
hist.col	the colour of lines of the histogram, if plot.hist=TRUE
hist.fill	the colour of the histogram, if plot.hist=TRUE
dens.fill	the color of the density plot, if plot.hist=TRUE
plot.hist	whether to use y_dots() or y_hist() for the continuous variables
plot	whether to plot
hist	whether histogram or dot plot
max.levels	excludes from plotting bucket plots for variables with less than max.levels, distinct values
title	title of the plot
percentage	if set, i.e. 0.50, plots a portotion of data otherwise for big data sets greater than 50.000 observartions it plots a porpotion
seed	the set.seed() argument
print.info	whether to print infomation when cutting the data usinf data_cut()
text.x.angle	how the text in the x-axis is printed (helping if say factors have a lot of levels). It can be a signle number of a vector. In both cases it will expand as a vector with length the number of explanatory variables. Therefore for full control a vector of the same length as the number of x-variables should be given.)
...	other arguments

Details

The function data_xyplot() it takes a data frame and plot all the explanarory variables against the response.

The function data_plot() it takes a data frame and plot all variables against the response. The continuous are plotted using y_dots() or y_hist() while the factors and integer as bar plots.

Value

Plots of the data

Author(s)

Mikis Stasinopoulos

References

Rigby, R. A. and Stasinopoulos D. M. (2005). Generalized additive models for location, scale and shape,(with discussion), *Appl. Statist.*, **54**, part 3, pp 507-554.

Rigby, R. A., Stasinopoulos, D. M., Heller, G. Z., and De Bastiani, F. (2019) *Distributions for modeling location, scale, and shape: Using GAMLSS in R*, Chapman and Hall/CRC. An older version can be found in <https://www.gamlss.com/>.

Stasinopoulos D. M., Rigby R.A. (2007) Generalized additive models for location scale and shape (GAMLSS) in R. *Journal of Statistical Software*, Vol. **23**, Issue 7, Dec 2007, <https://www.jstatsoft.org/v23/i07/>.

Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC.

Stasinopoulos, M.D., Kneib, T., Klein, N., Mayr, A. and Heller, G.Z., (2024). *Generalized Additive Models for Location, Scale and Shape: A Distributional Regression Approach, with Applications* (Vol. **56**). Cambridge University Press.

(see also <https://www.gamlss.com/>).

See Also

[data_names](#)

Examples

```
da <- rent99[,-2]
data_xyplot(da, rent)
data_plot(da)
y_zscores(da$rent)
data_response(da, response=rent)
```

Index

- * **~regression**
 - data_factor, 9
- * **diagnostic**
 - data_void, 27
- * **package**
 - gamlss.prepdata-package, 2
- * **regression**
 - cor_perm_test, 3
 - data_cor, 5
 - data_dim, 7
 - data_inter, 11
 - data_leverage, 13
 - data_mcor, 14
 - data_part, 19
 - data_rm, 21
 - data_scale, 23
 - data_str, 25
 - data_void, 27
 - data_xyplot, 29

ACE (data_mcor), 14

cor_boot (cor_perm_test), 3

cor_perm_test, 3

data_association (data_cor), 5

data_boot_index (data_part), 19

data_boot_weights (data_part), 19

data_bucket (data_xyplot), 29

data_cha2fac (data_str), 25

data_cor, 5, 9, 12, 29

data_cut (data_part), 19

data_dim, 7, 27

data_distinct (data_str), 25

data_exclude_class (data_rm), 21

data_fac2num (data_str), 25

data_factor, 9

data_few2fac (data_str), 25

data_form2X (data_scale), 23

data_formulae (data_scale), 23

data_int2num (data_str), 25

data_inter, 11

data_Kfold_index (data_part), 19

data_Kfold_weights (data_part), 19

data_leverage, 13

data_mcor, 14

data_na_obs (data_dim), 7

data_na_vars (data_dim), 7

data_names, 19, 32

data_names (data_dim), 7

data_omit (data_dim), 7

data_only_continuous (data_rm), 21

data_outliers, 17

data_part, 19

data_part_list (data_part), 19

data_pcor (data_cor), 5

data_plot (data_xyplot), 29

data_rename (data_dim), 7

data_response (data_xyplot), 29

data_rm, 21

data_rm1val (data_rm), 21

data_rmNAvars (data_rm), 21

data_scale, 23

data_select (data_rm), 21

data_shorter_names (data_dim), 7

data_str, 10, 21, 25

data_vars2data (data_scale), 23

data_void, 27

data_xyplot, 29

data_zscores, 14, 25

data_zscores (data_xyplot), 29

gamlss, 3

gamlss.family, 3

gamlss.prepdata

- (gamlss.prepdata-package), 2

gamlss.prepdata-package, 2

high_val (data_cor), 5

`low_val (data_cor)`, [5](#)
`mcor (data_mcor)`, [14](#)
`void (data_void)`, [27](#)

`y_distinct (data_str)`, [25](#)
`y_factor (data_factor)`, [9](#)
`y_outliers (data_outliers)`, [17](#)
`y_outliers_both (data_outliers)`, [17](#)
`y_outliers_by (data_outliers)`, [17](#)
`y_outliers_loop (data_outliers)`, [17](#)
`y_outliers_z (data_outliers)`, [17](#)
`y_zscores (data_xyplot)`, [29](#)