

Package: mbadtools (via r-universe)

October 14, 2026

Title MBAD Data Analytics Functions

Version 0.0.3.1013

Description Some functions useful for my MBAD courses on Data Analytics. Loads packages from core packages list on library call Includes Residual plots, Pareto plots, and adding significance color to pvalues in a gt table. Includes Granger causality table and plot functions. Includes functions for handling VAR models with bvar tools and plotting.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0), tidyverse, tidymodels, conflicted, ggfortify, skimr, GGally, gt, patchwork, lmttest, ggh4x, bvar tools, vars, vip, pdp, pre

Imports methods, rlang (>= 1.1.4), magrittr (>= 2.0.3), broom, dplyr, ggplot2, parsnip, stringr, tibble, tidyr, tidyselect, car, stats, workflowsets

LazyData true

Suggests knitr, rmarkdown, testthat (>= 3.0.0), randomForest, pak

Remotes bgreenwell/vip

Enhances qcc

URL <https://github.com/bhartman2/mbadtools>

BugReports <https://github.com/bhartman2/mbadtools/Issues>

Config/roxygen2/version 8.0.0

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev make libharfbuzz-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libssl-dev libnode-dev libx11-dev zlib1g-dev

Repository <https://bhartman2.r-universe.dev>

Date/Publication 2026-10-14 04:51:48 UTC

RemoteUrl <https://github.com/bhartman2/mbadtools>

RemoteRef HEAD

RemoteSha 28be3a8db7969cbc463fec5622d5ae344e9b5916

Contents

.onAttach	3
bvarirf_to_varirf	3
core_unloaded	4
gg_partial_residual_plots	5
gg_pdp	6
gg_pdp3D	6
gg_residual_plots	7
gg_vip	7
ggvar_fevdplot	8
ggvar_forecastplot	9
ggvar_irf	10
ggvar_plot_stability	10
ggvar_plot_stability_break	11
ggvar_plot_stability_lines	12
GrangerPlot	12
GrangerTable	13
GrangerTestPvals	14
GrangerTestTune	15
gt_add_significance	15
Kleinsrule	16
lib_packages	17
moody	18
pareto_arrange_data	18
pareto_cutoff_line	19
pareto_pct_labels	20
pareto_plot	20
pareto_sec_axis	21
rpart_unscale_object	22
Tukey_test	22
vi_pre	23

Index	24
--------------	-----------

.onAttach	<i>.onAttach</i>
-----------	------------------

Description

.onAttach

Usage

```
.onAttach(libname, pkgname)
```

Arguments

libname	character, name of library
pkgname	character, name of package

Value

package startup message, list of newly loaded packages in printable grid

bvarirf_to_varirf	<i>bvarirf_to_varirf</i>
-------------------	--------------------------

Description

Transforms a bvarirf object from `bvartools::irf()` into a varirf skeleton object from `vars::irf()`, for `ggvar_irf()` to plot. The bvarirf object only contains one response at a time.

Usage

```
bvarirf_to_varirf(  
  bvarirf,  
  impulse,  
  response,  
  cumulative = TRUE,  
  ortho = TRUE,  
  boot = TRUE,  
  ci = 0.05,  
  runs = 100  
)
```

Arguments

bvarirf	a bvarirf object made by <code>bvartools::irf()</code>
impulse	character, the impulse column name
response	character, the response column name
cumulative	logical, default TRUE
ortho	logical, default TRUE
boot	logical, default TRUE
ci	numeric p-value for confidence interval; in (0,1), default .05
runs	integer, number of runs, default 100

Value

a varirf skeleton object good enough for `ggvar_irf` to plot.

Examples

```
# Load data
data("e1", package="bvartools")
e1 <- diff(log(e1)) * 100

# Generate model data
model <- bvartools::gen_var(e1, p = 2, deterministic = 2,
                           iterations = 100, burnin = 10)
# Number of iterations and burnin should be much higher.

# Add prior specifications
model <- bvartools::add_priors(model)

# Obtain posterior draws
object <- bvartools::draw_posterior(model)

# Calculate IR
ir <- bvartools::irf(object, impulse = "invest", response = "cons")

# Plot IR
X = bvarirf_to_varirf(ir, impulse="invest", response="cons")
ggvar_irf(X)
```

core_unloaded

core_unloaded

Description

core_unloaded

Usage

```
core_unloaded()
```

Value

a list of packages that have not been loaded yet

```
gg_partial_residual_plots
```

Partial Residual Plots

Description

Draw partial residual plots with ggplot methods. Automatically plots all driver partial plots in the model.

Usage

```
gg_partial_residual_plots(.data, new_data = NULL, ...)
```

Arguments

<code>.data</code>	an lm object
<code>new_data</code>	data set model was trained on if class is <code>_lm</code> ; default <code>NULL</code>
<code>...</code>	parameters to be passed to <code>patchwork::wrap_plots()</code>

Value

a patchwork plot layout containing the partial residual plots.

Examples

```
data(freeny, package="datasets")
fit = lm(y ~ ., data = freeny)
gg_partial_residual_plots(fit)
```

`gg_pdp`*gg_pdp*

Description

Plots partial dependence plot for a variable selected with pdp

Usage

```
gg_pdp(.data, ...)
```

Arguments

<code>.data</code>	a pdp object with variable selected
<code>...</code>	additional arguments to pass to <code>geom_point()</code>

Value

a ggplot

Examples

```
mpg_fit = lm(mpg ~ disp + hp, data=mtcars)
gg_pdp(pdp::partial(mpg_fit, pred.var="disp", train=mtcars,
                    type="regression"))
```

`gg_pdp3D`*gg_pdp3D*

Description

Creates a 3D partial dependence plot from data created from a tidyverse model by pdp

Usage

```
gg_pdp3D(.data)
```

Arguments

<code>.data</code>	a pdp object with two variables selected
--------------------	--

Value

a ggplot object

gg_residual_plots	<i>gg_residual_plots</i>
-------------------	--------------------------

Description

ggplot versions of common residual plots meant to mimic the `plot.lm` and `ggfortify::autoplot.lm` residual plots.

Usage

```
gg_residual_plots(.data, items = c(1:3, 7), new_data = NULL, ...)
```

Arguments

- `.data` an `lm()` regression object
- `items` a vector of integers from 0 to 9; which of 9 plots to select; default `c(1:3,7)`
- `new_data` data set model was trained on if class is `_lm`; default `NULL`
- `...` parameters to be passed to `patchwork::wrap_plots()`

Value

a patchwork array of ggplot objects

Examples

```
data(freeny, package="datasets")
fit = lm(y ~ ., data=freeny)
gg_residual_plots(fit)
```

gg_vip	<i>gg_vip</i>
--------	---------------

Description

`gg_vip`

Usage

```
gg_vip(.data, ...)
```

Arguments

- `.data` a data frame created by `vip::vi`, with column names `c("Variable", "Importance")`
- `...` arguments to pass to `ggplot2::geom_col`

Value

a ggplot

Examples

```
mpg_fit = lm(mpg ~ ., data=mtcars)
gg_vip(vip::vi(mpg_fit))
```

ggvar_fevdplot	<i>ggvar_fevdplot</i>
----------------	-----------------------

Description

Forecast Error Variance Decomposition plots for `bvartools::bvar` objects from [bvartools](#); plots all responses or any one

Usage

```
ggvar_fevdplot(bvarobject, type = "oir", ...)
```

Arguments

<code>bvarobject</code>	a <code>bvarest</code> object from <code>bvartools</code>
<code>type</code>	character, type of fevd to create, no default, allowed types are <code>c("oir", "gir", "sir", "sgir")</code> ; only "oir" has sum to unity.
<code>...</code>	other parameters for <code>bvartools::fevd.bvar()</code> ; see documentation for details.

Value

a named list of ggplots, named by response variables of model. Display with `patchwork` or just plot one list member.

Examples

```
#' # Load data
library(bvartools)
data("e1")
e1 <- diff(log(e1)) * 100

# Generate model data
model <- bvartools::gen_var(e1, p = 2, deterministic = 2,
                           iterations = 100, burnin = 10)

model <- bvartools::add_priors(model)

# Obtain posterior draws
object <- bvartools::draw_posterior(model)
```

```
ggvar_fevdplot(object, type="oir", n.ahead=7)
```

```
ggvar_forecastplot      ggvar_forecastplot
```

Description

plot multiple timeseries and forecasts made with `predict`; use `design` and `patchwork::wrap_plots` to plot from list.

Usage

```
ggvar_forecastplot(bvar_pred, trun = 1)
```

Arguments

<code>bvar_pred</code>	predictions of class <code>bvarprd</code> from <code>bvartools::predict.bvar()</code> or <code>varprd</code> from <code>vars::predict()</code> .
<code>trun</code>	integer, start index of data to plot for all predictions; default 1

Value

a list of ggplots

Examples

```
# Load data
require(bvartools)
data("e1")
e1 <- diff(log(e1)) * 100
# Generate model data
model <- bvartools::gen_var(e1, p = 2, deterministic = 2,
                           iterations = 100, burnin = 10)
model <- bvartools::add_priors(model)
# Obtain posterior draws
object <- draw_posterior(model)
# Calculate forecasts
pred <- predict(object, new_d = rep(1, 10))
# Plot forecasts
ggvar_forecastplot(pred)
```

ggvar_irf

*ggvar_irf***Description**

Plots impulse response functions from a `vars::varirf` object

Usage

```
ggvar_irf(varirf)
```

Arguments

`varirf` a `varirf` object created using the `vars` package

Value

a `ggplot`

Examples

```
data("Canada", package="vars")
var.2c <- vars::VAR(Canada, p = 2, type = "const")
IRF = vars::irf(var.2c, impulse = "e", response = "prod")
ggvar_irf(IRF)
```

ggvar_plot_stability

*ggvar_plot_stability***Description**

Plots the list of empirical fluctuation processes created by the [stability](#) test; one is created for each equation. Use [wrap_plots](#) to display, where `n` is the number of plots. It wraps [ggvar_plot_stability_break](#) and [ggvar_plot_stability_lines](#)

If you want one or just some plots, make them into a list first

Usage

```
ggvar_plot_stability(varstabil, lines = TRUE, breaks = TRUE, a = 1.2516)
```

Arguments

`varstabil` a `varstabil` list object from [stability](#) or a list of [efp](#) objects
`lines` logical, plot the boundary lines? default TRUE
`breaks` logical, plot vertical line at break and value, default TRUE
`a` numeric, limit for boundary lines, default 1.2516; see [ggvar_plot_stability_break](#) for rationale

Value

a list of ggplots one for each variable

Examples

```
data("Canada", package="vars")
var.2c <- vars::VAR(Canada, p = 2, type = "const")
stab = vars::stability(var.2c)
ggvar_plot_stability(stab, lines=FALSE, breaks=FALSE) # override defaults for generic plot
```

ggvar_plot_stability_break
ggvar_plot_stability_break

Description

Internal function that adds layers to a [ggvar_plot_stability](#) plot; a vertical line and a text notation of the max value, at the max of the stability index. Called by [ggvar_plot_stability](#) optionally.

Usage

```
ggvar_plot_stability_break(pi, bv)
```

Arguments

pi	breakpoint index; obtain from process maximum point
bv	breakpoint value; use pi to obtain from process

Value

ggplot layers to add to a `ggvar_plot_stability()` plot

Examples

```
data("Canada", package="vars")
var.2c <- vars::VAR(Canada, p = 2, type = "const")
stab = vars::stability(var.2c)
ggvar_plot_stability(stab, lines=FALSE) # override default
```

```
ggvar_plot_stability_lines
      ggvar_plot_stability_lines
```

Description

Internal function that adds layers to a `ggvar_plot_stability` plot; adds horizontal boundary lines. Called by `ggvar_plot_stability` optionally.

Usage

```
ggvar_plot_stability_lines(a = 1.2516)
```

Arguments

`a` limit for upper and lower bounds; default 1.2516, taken from Gemini discussion of limits for [efp](#).

Value

a ggplot

Examples

```
data("Canada", package="vars")
var.2c <- vars::VAR(Canada, p = 2, type = "const")
stab = vars::stability(var.2c)
ggvar_plot_stability(stab, breaks=FALSE) # lines=T is default
```

GrangerPlot

GrangerPlot

Description

Creates list of plots of Granger test p-values for multiple lags from dataframe created by `GrangerTable`. Use this function on a data frame if the data column observations were taken at the same times

Usage

```
GrangerPlot(.data)
```

Arguments

`.data` a dataframe object created by `GrangerTable`.

Value

a list of two plots for each direction of Granger Causality test

See Also

[Examples](#), [grangertest](#)

Examples

```
## Not run:
data(moody, package="mbadtools")
patchwork::wrap_plots(
  GrangerPlot(
    GrangerTable(moody, ICS, Wgrowth)
  ),nrow=2)

## End(Not run)
```

GrangerTable

GrangerTable

Description

Computes p.values for Granger cause test of two time series, in both directions. Can be supplied as input to `mbadtools::GrangerPlot()`. Use this if the data is in a dataframe and you are sure the two series observations are taken at the same times.

Usage

```
GrangerTable(.data, s1, s2, max.lags = 10)
```

Arguments

<code>.data</code>	dataframe containing two numeric columns
<code>s1</code>	series 1 numeric column name
<code>s2</code>	series 2 numeric column name
<code>max.lags</code>	integer, maximum number of lags to consider, default =10.

Value

data frame with lags and p.values of two granger tests

See Also

[Examples](#), [grangertest](#)

Examples

```
data(moody, package="mbadtools")
moody %>% GrangerTable(ICS, Wgrowth)
```

GrangerTestPvals	<i>GrangerTestPvals</i>
------------------	-------------------------

Description

Use this function for two time series that were taken at the same times.

Usage

```
GrangerTestPvals(ts1, ts2, max.lags = 10)
```

Arguments

ts1	time series of type ts
ts2	time series of type ts
max.lags	number of lags to consider, default = 10

Details

Creates a data frame of p-values for bi-directional Granger Causality test for multiple lags, using `lmtest::grangertest`.

Value

data frame with `max.lags` rows and three columns: lag, the lag number, p12, the p-value for ts1->ts2, and p21, the reverse.

Examples

```
data(EuStockMarkets, package="datasets")
# extract two time series
DAX = EuStockMarkets[, "DAX"]
FTSE = EuStockMarkets[, "FTSE"]
# create the p-values table for the two directions
GrangerTestPvals(DAX, FTSE)
```

GrangerTestTune	<i>GrangerTestTune</i>
-----------------	------------------------

Description

Run a Granger Causality test in both directions for one time series versus another, over a range of different orders. Use this function for two time series that were taken at the same times.

Usage

```
GrangerTestTune(ts1, ts2, max.lags = 10)
```

Arguments

ts1	time series
ts2	time series
max.lags	maximum number of lags to try, default 10

Value

a list containing two ggplot objects, showing variation in p-value of Granger causality test in each direction with number of lags; can be plotted by new('ggmultiplot') or other means.

Examples

```
## Not run:
data(EuStockMarkets, package="datasets")
# extract two time series
DA = EuStockMarkets[, "DAX"]
FT = EuStockMarkets[, "FTSE"]
# plot the two plots with patchwork
GrangerTestTune(DA, FT) %>% patchwork::wrap_plots(nrow=2)

## End(Not run)
```

gt_add_significance	<i>gt_add_significance</i>
---------------------	----------------------------

Description

Makes a gt table, and adds colored significance notation, to a data frame containing p.value column. Useful for glance and tidy output from lm models.

Usage

```
gt_add_significance(.data, level = 0.05, digits = 4)
```

Arguments

<code>.data</code>	A tidy or glance tibble resulting from an <code>lm</code> model.
<code>level</code>	The significance level selected. default = .05
<code>digits</code>	number of digits to retain in numeric columns; default = 4

Value

a gt table with the pvalue column colored where result is significant.

Examples

```
## Not run:
data(cars)
fit = lm(dist~speed, data=cars);
gt_add_significance(broom::tidy(fit))

## End(Not run)
```

Kleinsrule

Kleinsrule

Description

Implements Klein's rule of thumb for linear models using `lm` to detect multicollinearity.

Usage

```
Kleinsrule(.data, response)
```

Arguments

<code>.data</code>	a dataframe with the target and drivers under study.
<code>response</code>	response for the regression

Details

Klein's rule of thumb is to run auxiliary regressions with each driver variable against all the others. Then compare these r-squareds to the full regression on all the drivers. The function builds a tibble to make this comparison easy.

Value

a tibble with the r-squared values of the main and all auxiliary regressions

Examples

```
## Not run:
data(longley, package="datasets")
# this data is known to be collinear
Kleinsrule(longley, response = "Employed")

## End(Not run)
```

`lib_packages`*lib_packages*

Description

enables packages listed in a character vector to an R script

Usage

```
lib_packages(pkgtxtvector, quiet = FALSE)
```

Arguments

<code>pkgtxtvector</code>	character vector of package names
<code>quiet</code>	logical suppress startup messages; default = TRUE

Value

same as [library](#)

See Also

[library](#), [suppressPackageStartupMessages](#)

Examples

```
## Not run:
pkgs = c("tidyverse", "car")
lib_packages(pkgs)

## End(Not run)
```

moody	<i>moody</i>
-------	--------------

Description

A dataset containing wage growth and consumer sentiment data.

Usage

moody

Format

A data frame with 299 rows and 3 columns:

- Date** date-time of observation
- Wgrowth** numeric, wage growth at that time
- ICS** numeric consumer sentiment survey at that time

Source

consumer sentiment data from University of Michigan Survey; Wage growth from Atlanta Fed Wage Tracker.

pareto_arrange_data	<i>pareto_arrange_data</i>
---------------------	----------------------------

Description

Arrange compute data for a pareto plot

Usage

pareto_arrange_data(.data, countcol)

Arguments

- .data data frame with a category variable and a count column, often produced with `dplyr::summarize`
- countcol name of the column with the counts, no quotes required

Value

a dataframe with the category and count columns plus the Cumulative and Percentage columns.

See Also

[Pareto Plot Examples](#), [pareto.chart](#)

Examples

```
data(chickwts, package="datasets")
K = chickwts %>% group_by(feed) %>% summarize(avwt = mean(weight))
K1 = pareto_arrange_data(K, avwt)
print(K1)
```

pareto_cutoff_line	<i>pareto_cutoff_line</i>
--------------------	---------------------------

Description

Puts a cutoff line on a pareto_plot

Usage

```
pareto_cutoff_line(maxcount, percent = 80)
```

Arguments

maxcount	maximum count
percent	cutoff percent, default=80

Value

added ggplot layer with the Pareto percentage line

See Also

[Pareto Plot Examples](#), [pareto.chart](#)

Examples

```
data(chickwts, package="datasets")
K = chickwts %>% group_by(feed) %>% summarize(avwt = mean(weight))
K1 = pareto_arrange_data(K, avwt)
pareto_plot(K1, feed, avwt) +
  pareto_cutoff_line(max(K1$avwt))
```

pareto_pct_labels	<i>pareto_pct_labels</i>
-------------------	--------------------------

Description

Add percent labels on cumulative percent line. Use additional parameters to size and position the labels on the plot. Data object must have a Percentage column.

Usage

```
pareto_pct_labels(maxcount, ...)
```

Arguments

maxcount	maximum count
...	additional parameters to geom_text

Value

added ggplot layer

See Also

[Pareto Plot Examples](#), [pareto.chart](#)

Examples

```
data(chickwts, package="datasets")
K = chickwts %>% group_by(feed) %>% summarize(avwt = mean(weight))
K1 = pareto_arrange_data(K, avwt)
pareto_plot(K1, feed, avwt) +
  pareto_pct_labels(max(K1$avwt), nudge_y=1, size=3)
```

pareto_plot	<i>pareto_plot</i>
-------------	--------------------

Description

Make a pareto plot from data arranged with pareto_arrange

Usage

```
pareto_plot(.data, category, count)
```

Arguments

.data	a pareto_arranged data frame
category	name of the category column
count	name of the count column

Value

a ggplot object

See Also

[Pareto Plot Examples](#), [pareto.chart](#)

Examples

```
data(chickwts, package="datasets")
K = chickwts %>% group_by(feed) %>% summarize(avwt = mean(weight))
K1 = pareto_arrange_data(K, avwt)
pareto_plot(K1, feed, avwt)
```

pareto_sec_axis	<i>pareto_sec_axis</i>
-----------------	------------------------

Description

pareto_sec_axis

Usage

```
pareto_sec_axis(maxcount)
```

Arguments

maxcount	numeric, maximum value of the count column
----------	--

Value

a ggplot layer

See Also

[Pareto Plot Examples](#), [pareto.chart](#)

Examples

```
data(chickwts, package="datasets")
K = chickwts %>% group_by(feed) %>% summarize(avwt = mean(weight))
K1 = pareto_arrange_data(K, avwt)
pareto_plot(K1, feed, avwt) +
  pareto_sec_axis(max(K1$avwt))
```

rpart_unscale_object	<i>rpart_unscale_object</i>
----------------------	-----------------------------

Description

Rescales the driver splits in an `rpart` object using means and sds of training data. Caution: the new object only has splits scaled that are used in the tree plot and rules.

Usage

```
rpart_unscale_object(surr_model_object, train_data_p)
```

Arguments

- `surr_model_object`
an `rpart` model object
- `train_data_p` training data, with unscaled drivers

Value

`rpart` object with splits index column rescaled

Tukey_test	<i>Tukey_test</i>
------------	-------------------

Description

Tukey test of curvature in linear regression models

Usage

```
Tukey_test(.data, fmt = c("gt", "summary"))
```

Arguments

- `.data` an `lm()` regression object
- `fmt` character `c("gt", "summary")`, default `"gt"`

Value

a list; summary a data frame with the variables, statistics and p-values gt a gt table with the same

Examples

```
fit = lm(mpg~hp, data=mtcars)
Tukey_test(fit, fmt = "summary")
Tukey_test(fit)
```

vi_pre	vi_pre
--------	--------

Description

Creates vip:vi() style data frame for a model of class pre Features are sorted in descending order of importance. Models of this class are essentially linear so a sign is reported as well. The result can be plotted with gg_vip.

Usage

```
vi_pre(model, num_features = 15)
```

Arguments

model	model generated by pre::pre
num_features	number of features to retain; default 15

Value

a data frame with num_features rows and several columns:

- code, rule id; created by pre
- Variable, character variable; shows the rule in English
- Importance, numeric variable importance
- coefficient, numeric; coefficient in linear (lasso) regression
- sd, numeric; standard deviation of coefficient
- Sign, factor; -1 or 1 depending on sign of coefficient

Index

* datasets

moody, [18](#)

.onAttach, [3](#)

bvarirf_to_varirf, [3](#)

bvartools, [8](#)

bvartools::irf(), [3](#), [4](#)

core_unloaded, [4](#)

efp, [10](#), [12](#)

gg_partial_residual_plots, [5](#)

gg_pdp, [6](#)

gg_pdp3D, [6](#)

gg_residual_plots, [7](#)

gg_vip, [7](#)

ggvar_fevdplot, [8](#)

ggvar_forecastplot, [9](#)

ggvar_irf, [4](#), [10](#)

ggvar_irf(), [3](#)

ggvar_plot_stability, [10](#), [11](#), [12](#)

ggvar_plot_stability_break, [10](#), [11](#)

ggvar_plot_stability_lines, [10](#), [12](#)

GrangerPlot, [12](#)

GrangerTable, [13](#)

grangertest, [13](#)

GrangerTestPvals, [14](#)

GrangerTestTune, [15](#)

gt_add_significance, [15](#)

Kleinsrule, [16](#)

lib_packages, [17](#)

library, [17](#)

moody, [18](#)

pareto.chart, [19–21](#)

pareto_arrange_data, [18](#)

pareto_cutoff_line, [19](#)

pareto_pct_labels, [20](#)

pareto_plot, [20](#)

pareto_sec_axis, [21](#)

rpart_unscale_object, [22](#)

stability, [10](#)

suppressPackageStartupMessages, [17](#)

Tukey_test, [22](#)

vars::irf(), [3](#)

vi_pre, [23](#)

wrap_plots, [10](#)