Package ‘cocoreg’

May 30, 2017

Type Package

Title Extract Shared Variation in Collections of Data Sets Using Regression Models

Version 0.1.1

Description
The algorithm extracts shared variation from a collection of data sets using regression models.

Author Jussi Korpela [aut, cre], Andreas Henelius [aut], Finnish Institute of Occupational Health [cph]

Maintainer Jussi Korpela <jussi.korpela@iki.fi>

Depends R (>= 3.2.0)

Imports ggplot2, gridExtra, CCAGFA, RGCCA, combinat, abind, MASS, glmnet, pls, e1071, multiway, reshape

License MIT + file LICENSE

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-05-30 10:36:05 UTC

R topics documented:

`add_notches` *Add notch-like gaussian snippets to an existing signal x*

Description

Add notch-like gaussian snippets to an existing signal x

Usage

```
add_notches(x, pos, sd = 0.01 * length(x), amplitude = 1)
```

Arguments

`x`	[1,N] numeric, Original data
`pos`	[1,m] integer, Positions to add notches to
`sd`	[1,1] numeric, (optional) Desired width of the Gaussian notch
`amplitude`	[1,1] numeric, (optional) Desired amplitude for the notches

Value

1,N numeric, Modified signal with notches

apply_dc_meta *Apply extracted properties of a data collection to a data collection (restore)*

Description

Apply extracted properties of a data collection to a data collection (restore)

Usage

```
apply_dc_meta(df_list, meta)
```

Arguments

`df_list`	list of data.frames, The data collection to process
`meta,`	list, Output of get_dc_meta()

Value

A list of data.frames, the data collection with updated metadata

average_R2_dflst *Computes the R^2 (variance explained) between two lists of data.frames*

Description

Computes the R^2 (variance explained) between two lists of data.frames

Usage

```
average_R2_dflst(df_orig_lst, df_est_lst)
```

Arguments

`df_orig_lst`	List of original data.frames
`df_est_lst`	List of estimated data.frames

Value

Returns a data.frame with R2 values, one value for each data set and variable. Molten/long format.

BGFA_cocoreg_interface

Apply GFA using the same interface as cocoreg()

Description

Note: if K is too high GFA() might not converge in a meaningful time or the computation may mysteriously crash.

Usage

```
BGFA_cocoreg_interface(df_list, K = 8, Nrep = 2, threshold = 0.001)
```

Arguments

`df_list`	[1,m] list of data.frames, Input data to GFA in COCOREG format
`K`	[1,1] int, (Maximum) number of GFA components
`Nrep`	[1,1] int, Number of random initialization used for learning the model
`threshold`	[1,1] double, GFA model trimming threshold

Value

A list with elements:

`$data:`	[1,m] list of data.frames, Original data reconstructed using only those latent components that are active in all datasets
`$model:`	a list, Non-trimmed output of CCAGFA::GFA()
`$dataid:`	string, Dataset identifier string
`$method:`	string, Analysis method identifier string
`$wall_time_taken:`	[1,1] double, Time taken to run the analysis in seconds

BGFA_joint_info

Project BGFA components common to all datasets back to the original space

Description

Project BGFA components common to all datasets back to the original space

Usage

```
BGFA_joint_info(model, threshold = 0.001)
```

Arguments

`model`	Output of CCAGFA::GFA()
`threshold`	[1,1] double, GFA model trimming threshold

Value

A list of data.frames, Original data reconstructed using only those latent components that are active in all datasets

---

`cocoreg` *The Common Components by Regression (CoCoReg) algorithm*

Description

An algorithm that extracts common variation between datasets using regression.

Usage

```
cocoreg(data, cyclic = FALSE, mapping_function = mapping_lm,
  sample_paths = FALSE, center_data = T, scale_data = F)
```

Arguments

`data`	[1,K] list of data.frames.
`cyclic`	boolean, Operation mode: cyclic or non-cyclic
`mapping_function`	function, The function to use in mappings. See mapping_lm() for an example.
`sample_paths`	boolean, If FALSE all paths are computed. If TRUE a subset of paths is taken: one (random) path for each starting point. Currently implemented only for cyclic=F.
`center_data`	boolean, Should the data be centered?
`scale_data`	boolean, Should the data be scaled?

Value

A list with elements:

`$data:`	[1,K] list of data.frames containing the joint information, organised identically to the input data.
`$mappings:`	[1,K*K-K] list of functions, mappings between datasets
`$paths:`	[(K-1)(K-2)!, K] list of lists, paths for each data set
`$cyclic:`	input cyclic as is
`$sample_paths:`	boolean, TRUE if paths have been sampled, FALSE otherwise.
`$dataid:`	string, Dataset identifier string
`$method:`	string, Analysis method identifier string
`$wall_time_taken:`	[1,1] double, Time taken to run the analysis in seconds

Examples

```
dc <- create_syn_data_toy()
ccr <- cocoreg(dc$data)

ggplot_dflst(dc$data, ncol=1)
ggplot_dflst(ccr$data, ncol=1)

## Not run:
ggplot_dclst(list(orig = dc$data, ccr = ccr$data))
ggplot_dclst(list(orig = dc$data, shared = ccr$data), legendMode = 'none')
ggplot_dclst(list(orig = dc$data, ccr = ccr$data), legendMode = 'all')

## End(Not run)
```

`cocoreg_by_path` *Compute D_joint for dataset i separately for all paths Can be used to study path variability*

Description

Compute D_joint for dataset i separately for all paths Can be used to study path variability

Usage

```
cocoreg_by_path(data_orig, ccr, ds_ind)
```

Arguments

`data_orig`	list of data.frames, Original data collection
`ccr`	list, output of cocoreg(data_orig)
`ds_ind`	integer, index of the dataset to process

Value

A list of data.frames, D_joint corresponding to each path that ends at 'ds_ind' i.e. paths defined by ccr$paths[,ds_ind]. Dimensions of the matrices are the same as for the data.frames in data_orig.

Examples

```
## Not run:
ccr <- cocoreg(data_list)
jibp <- cocoreg_by_path(ccr, 1)

## End(Not run)
```

compose — *Calculate the composition formed by applying all functions in the given path to a dataset.*

Description

Calculate the composition formed by applying all functions in the given path to a dataset.

Usage

```
compose(x, path, mappings)
```

Arguments

`x`	A data frame or vector
`path`	A list describing the path.
`mappings`	A list containing the mapping functions described in the path. Usually a list of all M*M-M available mappings between the M data sets.

Value

A vector containing the result of the composition.

compose_all — *Calculate the average of the composition formed by applying all functions in all possible paths to a dataset.*

Description

Calculate the average of the composition formed by applying all functions in all possible paths to a dataset.

Usage

```
compose_all(x, paths, mappings)
```

Arguments

`x`	A list of data frames.
`paths`	A list of list with paths.
`mappings`	A list containing the mapping functions described in the path.

Value

A vector containing the average composition from all the paths.

`create_mappings` *Generate all possible pairwise mappings between the given multivariate datasets.*

Description

The following naming convention is used in the output: '1-2' means '1' mapped to '2', i.e.,, '2' explained by '1'.

Usage

```
create_mappings(data, mapping_function = mapping_lm)
```

Arguments

`data`	A list of data.frames (the datasets)
`mapping_function`	(optional) Default is `mapping_lm`.

Value

A named list containing the pairwise mapping functions.

`create_syndata_mv` *Create multivariate synthetic data*

Description

Create multivariate synthetic data

Usage

```
create_syndata_mv(Z, W, max_var_arr = rep(1, length(W)))
```

Arguments

`Z`	[N,L] matrix, Latent factors, N observations, L factors
`W`	a [1,K] list of [L,D_k] matrices or [L,D,K] array, Projections from latent factors to data, D_k variables per dataset
`max_var_arr`	(optional) [1,K] numeric, Relative maximum amplitude of noise

Value

A list with elements:

`data`	Data collection as a list of data.frames
`Z`	Signals used
`W`	Mixing matrix used
`E`	Noise
`var.coef`	Noise multiplication factor used

Each dataset is a data.frame to gain compatibility with lm() and glm()

`create_syndata_pwl` *A non-linear data collection using piecewise linearity*

Description

A non-linear data collection using piecewise linearity

Usage

```
create_syndata_pwl()
```

Value

A list with elements

`data`	Data collection as a list of data.frames
`Z`	Signals used
`W`	Mixing matrix used
`Z_all`	Signals shared by all datasets in the collection
`Z_sub`	Signals not shared by all datasets
`W_all`	Mixing weights for Z_all
`W_sub`	Mixing weights for Z_sub as a list of matrices, one matrix per dataset
`E`	Noise
`var.coef`	Noise multiplication factor used

`create_syn_data_puvar` *A data collection with variables that "become unrelated during measurement"*

Description

A data collection with variables that "become unrelated during measurement"

Usage

```
create_syn_data_puvar()
```

Value

A list with elements

`data`	Data collection as a list of data.frames
`Z_all`	Signals shared by all datasets in the collection
`Z_sub`	Signals not shared by all datasets
`W_all`	Mixing weights for Z_all
`W_sub`	Mixing weights for Z_sub
`E`	Noise
`var.coef`	Noise multiplication factor used

`create_syn_data_toy` *An illustrative synthetic data collection*

Description

Model: D_k = D_shared_by_all + D_shared_by_subset + D_unique,

Usage

```
create_syn_data_toy(N = 100, normalize = T, noisemf = 0.1)
```

Arguments

`N`	Number of observations in data as integer
`normalize`	(optional) Should the data be processed with dl_scale()? A boolean value.
`noisemf`	(optional) Multiplication factor for noise

Value

A list with elements

`data`	Data collection as a list of data.frames
`Z_all`	Signals shared by all datasets in the collection
`Z_sub`	Signals not shared by all datasets
`W_all`	Mixing weights for Z_all
`W_sub`	Mixing weights for Z_sub
`E`	Noise
`var.coef`	Noise multiplication factor used

Examples

```
## Not run:
dc <- create_syn_data_toy()
ggplot_dflst(dc$data, ncol = 1)

## End(Not run)
```

`create_syn_data_uds` *A data collection with one unrelated dataset*

Description

A data collection with one unrelated dataset

Usage

```
create_syn_data_uds()
```

Value

A list with elements

`data`	Data collection as a list of data.frames
`Z_all`	Signals shared by all datasets in the collection
`Z_sub`	Signals not shared by all datasets
`W_all`	Mixing weights for Z_all
`W_sub`	Mixing weights for Z_sub
`E`	Noise
`var.coef`	Noise multiplication factor used

`create_syn_data_uvar` *A collection with unrelated variables*

Description

A collection with unrelated variables

Usage

```
create_syn_data_uvar()
```

Value

A list with elements

`data`	Data collection as a list of data.frames
`Z_all`	Signals shared by all datasets in the collection
`Z_sub`	Signals not shared by all datasets
`W_all`	Mixing weights for Z_all
`W_sub`	Mixing weights for Z_sub
`E`	Noise
`var.coef`	Noise multiplication factor used

`create_Z_linear` *Contains functions to create synthetic datasets with different properties. The create_syn_data_*() functions follow the scheme: "total variation = shared_by_all + shared_by_subset + noise" Create signals*

Description

Contains functions to create synthetic datasets with different properties. The create_syn_data_*() functions follow the scheme: "total variation = shared_by_all + shared_by_subset + noise" Create signals

Usage

```
create_Z_linear(N, decorrelate = T)
```

Arguments

`N`	Number of observations in data as integer
`decorrelate`	(optional) Should the variables be de-correlated?

Value

A [N,3] matrix of signals

`cshift` *Circularly shift vector elements*

Description

Circularly shift vector elements

Usage

```
cshift(x, by)
```

Arguments

`x`	[1,N] numeric, A vector
`by`	[1,1] integer, How many positions to shift. by > 0 -> shift to right by = 0 -> no shift by < 0 -> shift to left

Value

1,N numeric, Circularly shifted signal

`data_collections2ggdf` *Catenate a set of data collections (lists of data.frames) into a single molted data.frame.*

Description

Can be used e.g. to prepare data for plotting with ggplot().

Usage

```
data_collections2ggdf(..., id.vars = NULL)
```

Arguments

`...`	Several lists of data.frames to catenate
`id.vars`	[1,m] string, ID variables for reshape::melt

Value

A data.frame with elements of ... melted and catenated vertically into a single data.frame.

Extra columns created:

`ds:`	dataset id within data collection
`dc:`	data collection id

Examples

```
df_lst <- list(df1 = iris[,2:3], df2 = iris[2:3])
data_collections2ggdf(dc1 = df_lst, dc2 = df_lst)
```

`data_matrix_rmse` *Compute RMSE between data.matrices dm1 and dm2*

Description

A data.matrix has observations as rows and variables as columns

Usage

```
data_matrix_rmse(dm1, dm2)
```

Arguments

`dm1`	[N,M] numeric, First data.matrix
`dm2`	[N,M] numeric, Second data.matrix

Value

1,M numeric, A vector of RMSE values, one per variable.

Examples

```
## Not run:
 dm1 <- matrix(rep(1,6),nrow=2)
 dm2 <- matrix(rep(3,6),nrow=2)
 data_matrix_rmse(dm1, dm2)

 first = list(dm1, dm1)
 second = list(dm2, dm2)
 (tmp = mapply(data_matrix_rmse, first, second, SIMPLIFY=FALSE))

## End(Not run)
```

`dc_variability` *Compute ds_variability for all datasets in a data collection*

Description

Compute ds_variability for all datasets in a data collection

Usage

```
dc_variability(data, ccr)
```

Arguments

`data`	Unprocessed dataset as a list of data.frames
`ccr`	Processed dataset as a list of data.frames, output of cocoreg()$data

Value

Path variability as a data.frame

`dflst2array` *Catenate a list of data.frames to a matrix along dim*

Description

Catenate a list of data.frames to a matrix along dim

Usage

```
dflst2array(df_lst, dim = 2)
```

Arguments

`df_lst`	[1,m] list of data.frames, A list of data.frames to process
`dim`	[1,1] int, Dimension to apply over

Value

A matrix with elements of df_lst converted to matrix and catenated along dim.

dflst2df — *Catenate a list of data.frames vertically to a single data.frame*

Description

Assumes equal variables for all datasets! Output has columns: <variables>, "dataset" Preserves list element names in column "dataset". For a more generic approach see dflst2dfmelt (uses reshape::melt(df_lst))

Usage

```
dflst2df(df_lst, id_var_name = "dataset")
```

Arguments

`df_lst`	[1,m] list of data.frames, A list of data.frames to process
`id_var_name`	string, Column name for the dataset id variable

Value

A data.frame with elements of df_lst catenated vertically, An extra column with dataset id is added.

dflst2dfmelt — *Combine a list of data.frames to a single molten data.frame*

Description

Output maximally "molten" with columns "dataset", "obs", "variable", "value" Preserves list element names in column "dataset".

Usage

```
dflst2dfmelt(df_lst)
```

Arguments

`df_lst`	[1,m] list of data.frames, A list of data.frames to process

Value

A data.frame with elements of df_lst combined using reshape::melt().

Extra columns:

dataset dataset name

obs running observation index (time)

`dflst_add_ds` *Add a data.frame (dataset) to a list of data.frames (datasets)*

Description

Add a data.frame (dataset) to a list of data.frames (datasets)

Usage

```
dflst_add_ds(dflst, df, dsname)
```

Arguments

`dflst`	[1,m] list of data.frames, A list of data.frames
`df`	data.frame, Data frame to add
`dsname`	string, Dataset name for the data.frame to add

Value

A list of data.frames, A new list of data.frames with one new dataset in the end

`dflst_dsnames2varnames` *Append dataset names to variable names of the respective dataset*

Description

Append dataset names to variable names of the respective dataset

Usage

```
dflst_dsnames2varnames(dflst, sep = "_")
```

Arguments

`dflst`	[1,m] list of data.frames, A list of data.frames to process
`sep`	string, Separator to use

Value

A [1,m] list of data.frames with modified variable names.

`dflst_pca` *Apply PCA to the data after catenating data.frames horizontally*

Description

Apply PCA to the data after catenating data.frames horizontally

Usage

```
dflst_pca(df_lst, center = F, scale = F)
```

Arguments

`df_lst`	[1,m] list of data.frames, A list of data.frames to process
`center`	boolean, TRUE -> center data, FALSE -> do nothing
`scale`	boolean, TRUE -> scale data, FALSE -> do nothing

Value

A list with elements:

`pcdf:`	data.frame, PCA components (prcomp()$x)
`model:`	list, Output of prcomp()

`df_ggplot_melt` *Melt data.frame into ggplottable format*

Description

Melts a data.frame into format that is suitable for use with ggplot2. Creates the time variable ’t’ used by plotting functions.

Usage

```
df_ggplot_melt(df)
```

Arguments

`df`	A data.frame

Value

A ggplot2 compatible data.frame with time variable

Examples

```
## Not run:
dc <- create_syn_data_toy()
df <- dc$data[[1]]
str(df)
str(df_ggplot_melt(df))

## End(Not run)
```

`df_scale` *Apply scale on a numeric data.frame*

Description

Apply scale on a numeric data.frame

Usage

```
df_scale(df, ...)
```

Arguments

`df`	data.frame, A numeric data.frame to process
`...`	arguments to scale()

Value

data.frame, A scaled data.frame with attributes preserved

`df_scale_ols` *Scales variables in data.frame dfx using ordinary least squares such*

Description

Scales variables in data.frame dfx using ordinary least squares such that the scaled result explains as much of the variance in dfy as possible. Scaling is done separately for each variable (i.e. no linear mixing of variables). Assumes data.frames dfx and dfy to be of identical structure. Intended use: to scale up cocoreg projections to account for the lost variance.

Usage

```
df_scale_ols(dfx, dfy)
```

Arguments

`dfx`	data.frame, Data frame to use as independent variable
`dfy`	data.frame, Data frames to use as dependent variable

Value

data.frame, A rescaled version of dfx with dimnames from dfy.

Examples

```
## Not run:
dc <- create_syn_data_toy()
ccr <- cocoreg(dc$data)
dfLst <- mapply(df_scale_ols, ccr$data, dc$data , SIMPLIFY=F)

## End(Not run)
```

`dl_remove_NA` *Remove rows with NA values from a list of data.frames*

Description

Same rows removed from all data frames in the list.

Usage

```
dl_remove_NA(df_lst)
```

Arguments

`df_lst`	[1,m] list of data.frames, A list of data.frames to process

Value

1,m list of data.frames, Data without rows that contain NA, same rows removed from all data.frames in the input list.

`dl_scale` *Run scale() on a list of data.frames*

Description

Run scale() on a list of data.frames

Usage

```
dl_scale(dl, ...)
```

Arguments

`dl`	A list of data.frames
`...`	Additional arguments to be passed on to scale

Value

A list of data.frames that have been processed using scale() and converted back to data.frame

`ds_variability` *Compute variability_components for a dataset*

Description

Note: might not work for cyclic ccr <TODO>

Usage

```
ds_variability(data, ccr, ds_ind)
```

Arguments

`data`	Unprocessed dataset as a list of data.frames
`ccr`	Processed dataset as a list of data.frames, output of cocoreg()$data
`ds_ind`	Starting dataset of the set of paths to analyze as [1,1] integer

Value

Path variability as a data.frame

`generate_mapping_function` *Generate a mapping function between two datasets*

Description

Generate a mapping function between two datasets, using some method, such as linear regression (lm), or some classifier such as a random forest (randomForest).

Wraps the function as well as data into a single object.

Usage

```
generate_mapping_function(method = lm)
```

Arguments

`method`	A function to be used in the mapping. A function object.

Value

Returns a function object that does the mapping between two datasets i.e. from dataset 1 to dataset 2.

`generate_paths` *Generate all/some paths between points*

Description

Generate all/some paths between points

Usage

```
generate_paths(ind, n, cyclic = FALSE, sample_paths = FALSE)
```

Arguments

`ind`	[1,2] int, The starting and ending point c(start, end).
`n`	[1,1] int, Number of points in the whole set.
`cyclic`	boolean, Should the path be cyclic (1-2-1) or noncyclic (2-1).
`sample_paths`	boolean, If FALSE, all possible paths are generated. If true one path per ending point is selected.

Value

A list of lists containing the paths.

`generate_paths_cyclic` *Generate cyclic paths*

Description

From a set of n numbers, generate all possible paths starting from and ending on a given number.

Usage

```
generate_paths_cyclic(ind, n)
```

Arguments

`ind`	[1,1] ind, The starting dataset (equals to ending dataset because of cycle).
`n`	[1,1] ind, The number of datasets.

Value

A list of lists containing the paths.

`generate_paths_noncyclic` *Generate non-cyclic paths*

Description

From a set of n numbers, generate all possible paths starting from and ending on a given number.

Usage

```
generate_paths_noncyclic(ind, n, sample_paths = FALSE)
```

Arguments

`ind`	The starting dataset
`n`	The number of datasets.
`sample_paths`	boolean, If FALSE, all possible paths are generated. If TRUE one path per ending point is selected.

Value

A list of lists containing the paths.

`get_dc_meta` *Extract important properties of data collection*

Description

Extract important properties of data collection

Usage

```
get_dc_meta(df_list, type = "current")
```

Arguments

`df_list`	list of data.frames, The data collection to process
`type`	string, If 'current' then data collection metadata is collected from the data collection itself. If 'original' then metadata is collected from special attributes.

Value

A list with elements:

`$dcnames:`	Dataset names
`$dcdimnames:`	A list of dataset dimension names for each dataset

`get_range_datalist` *Get [min, max] of a list of numeric objects*

Description

Get [min, max] of a list of numeric objects

Usage

```
get_range_datalist(dataList)
```

Arguments

`dataList`	[1,m] list of numeric objects

Value

1,2 double, [min, max] of the input

`get_starting_dataset` *Helper function to get the starting dataset based on a path.*

Description

Helper function to get the starting dataset based on a path.

Usage

```
get_starting_dataset(p)
```

Arguments

`p`	A path.

Value

A number indicating the starting dataset

`ggcompare_dclst` *Compare data collections variable by variable*

Description

Compare data collections variable by variable

Usage

```
ggcompare_dclst(dclst)
```

Arguments

`dclst`	A (named) list of data collections i.e. a list of lists of data.frames (see examples)

Value

Returns a ggplot object (which is by default printed if not assigned to variable)

ggplot_dclst *Plotting data collections using ggplot*

Description

Plotting data collections using ggplot

Usage

```
ggplot_dclst(dc_lst, ylim = NULL, titleArr = names(dc_lst),
  legendMode = "none", dfplot = ggplot_df)
```

Arguments

dc_lst	A list of data collections i.e. a list of lists of data.frames (see examples)
ylim	(optional) y-axis limits as [1,2] numeric, passed on to dfplot() as 'ylim'
titleArr	(optional) Plot column titles as [1, length(dc_lst)] string array
legendMode	(optional) Where to put legend, allowed values c('none','first','all')
dfplot	(optional) Function used to plot a data.frame (one panel in final plot)

Value

Produces a plot to the active graphics device

Examples

```
## Not run:
dc <- create_syn_data_toy()
ccr <- cocoreg(dc$data)
ggplot_dclst(list(d1 = dc$data, d2 = ccr$data, dn = dc$data))

## End(Not run)
```

ggplot_df *Plotting data.frame using ggplot*

Description

Plotting data.frame using ggplot

Usage

```
ggplot_df(df, titlestr = NULL, ylabstr = NULL, ylim = NULL,
  color = NULL, linetype = NULL, logy = F)
```

Arguments

`df`	A data.frame to plot
`titlestr`	(optional) Title of plot as string
`ylabstr`	(optional) Y-axis label as string
`ylim`	(optional) y-axis limits as [1,2] numeric, passed on to dfplot() as 'ylim'
`color`	(optional) Input for manual color scale
`linetype`	(optional) Input for manual linetype scale
`logy`	(optional) Should y-axis be logarithmic? A boolean value.

Value

Returns a ggplot2 object

Examples

```
## Not run:
dc <- create_syn_data_toy()
ggplot_df(dc$data[[1]])

## End(Not run)
```

`ggplot_dflst` *Plot a list of data.frames using ggplot2*

Description

Plot a list of data.frames using ggplot2

Usage

```
ggplot_dflst(dflst, ncol = 1, plot = T, plotfun = ggplot_df, ...)
```

Arguments

`dflst`	A list of datasets as a list of data.frames
`ncol`	(optional) Number of columns in final plot
`plot`	(optional) Plot or not: if TRUE produces a plot else returns a list of ggplot objects
`plotfun`	(optional) Function used to plot a data.frame (one panel in final plot)
`...`	(optional) Additional parameters passed on to plotfun

Value

Produces a plot to the active graphics device or returns a list of ggplot objects

Examples

```
## Not run:
dc <- create_syn_data_toy()
ggplot_dflst(dc$data)

## End(Not run)
```

make_data_gauss_2d *Make 2D gauss data (maybe obsolete)*

Description

Make 2D gauss data (maybe obsolete)

Usage

```
make_data_gauss_2d(n, var, angle_deg, scale = T, seed = 42)
```

Arguments

n	[1,1] int, Number of observations
var	[1,2] numeric, Variances
angle_deg	[1,1] numeric, Rotation angle
scale	boolean, Scale data? T -> scale, F -> do not scale
seed	[1,1] int, Random seed

Value

Matrix of 2D gaussian data

mappings_R2_matrix *Extract R2 values from a list of mappings using summary()*

Description

Extract R2 values from a list of mappings using summary()

Usage

```
mappings_R2_matrix(mappings, n_datasets, aggfun = mean)
```

Arguments

`mappings`	[M*M-M] list, Exhaustive list of mappings between the M datasets
`n_datasets`	integer, Number of datasets i.e. M
`aggfun`	function, A function to apply when aggregating R2 values over variables in a multivariate dataset

Value

A [M,M] matrix of R2 values stored such that the R2 value for mapping a->b is read from row a and column b.

Examples

```
## Not run:
ccr <- cocoreg(data_collection)
R2mat <- mappings_R2_matrix(ccr$mappings, length(ccr$data))

## End(Not run)
```

`mapping_glmnet` *Define a mapping function using glmnet::glmnet*

Description

Define a mapping function using glmnet::glmnet

Usage

```
mapping_glmnet(data1, data2)
```

Arguments

`data1`	data.frame, Dataset 1, the independent variables
`data2`	data.frame, Dataset 2, the dependent variables

Value

Returns a function object that does the mapping between two datasets.

`mapping_lm` *Mapping stats::lm*

Description

Mapping stats::lm

Usage

```
mapping_lm(data1, data2)
```

Arguments

`data1`	data.frame, Dataset 1, the independent variables
`data2`	data.frame, Dataset 2, the dependent variables

`mapping_lmridge` *Define a mapping function using MASS::lm.ridge*

Description

Define a mapping function using MASS::lm.ridge

Usage

```
mapping_lmridge(data1, data2)
```

Arguments

`data1`	data.frame, Dataset 1, the independent variables
`data2`	data.frame, Dataset 2, the dependent variables

Value

Returns a function object that does the mapping between two datasets.

`mapping_pcr` *Define a mapping function using pls::pcr*

Description

Define a mapping function using pls::pcr

Usage

```
mapping_pcr(data1, data2)
```

Arguments

`data1`	data.frame, Dataset 1, the independent variables
`data2`	data.frame, Dataset 2, the dependent variables

Value

Returns a function object that does the mapping between two datasets.

`mapping_rf` *Mapping randomForest*

Description

Mapping randomForest

Usage

```
mapping_rf(data1, data2)
```

Arguments

`data1`	data.frame, Dataset 1, the independent variables
`data2`	data.frame, Dataset 2, the dependent variables

`mapping_rlm` *Mapping MASS::rlm*

Description

Mapping MASS::rlm

Usage

```
mapping_rlm(data1, data2)
```

Arguments

`data1`	data.frame, Dataset 1, the independent variables
`data2`	data.frame, Dataset 2, the dependent variables

`mapping_svm` *Mapping svm*

Description

Mapping svm

Usage

```
mapping_svm(data1, data2)
```

Arguments

`data1`	data.frame, Dataset 1, the independent variables
`data2`	data.frame, Dataset 2, the dependent variables

`mapping_svm_sigmoid` *Mapping svm using sigmoid*

Description

Mapping svm using sigmoid

Usage

```
mapping_svm_sigmoid(data1, data2)
```

Arguments

`data1`	data.frame, Dataset 1, the independent variables
`data2`	data.frame, Dataset 2, the dependent variables

`matrix_variability` *Compute "variance" of the matrices using Frobenius norm. Variance is by default computed with respect to the mean of the matrices.*

Description

Compute "variance" of the matrices using Frobenius norm. Variance is by default computed with respect to the mean of the matrices.

Usage

```
matrix_variability(mat_lst, mean_mat = apply(abind::abind(mat_lst, along = 3),
  c(1, 2), mean))
```

Arguments

`mat_lst`	A [1,M] list of [I,J] matrices from which variability should be computed
`mean_mat`	A [I,J] matrix describing the mean observation for mat_lst.

Value

A list with elements

`fbnorm`	Frobenius norm values for each of the matrices
`diff`	[I,J,M] matrix of differences mat_lst-mean_mat

`nplst_reorder_grid` *Reorders a nested list of ggplots*

Description

Reorders a nested list of ggplots to ncol columns prior to calling grid.arrange() Note: p_list is a list of lists of ggplots. p_list = list(p_list1, p_list2,...)

Usage

```
nplst_reorder_grid(p_list, ncol)
```

Arguments

`p_list`	A list of lists of ggplots
`ncol`	Target number of columns, integer value

Value

A reordered and flattened version of input list as a list of ggplot2 objects

`PCA_cocoreg_interface` *PCA projection using cocoreg interface*

Description

PCA projection using cocoreg interface

Usage

```
PCA_cocoreg_interface(df_list, prc_th = 0.9)
```

Arguments

`df_list`	[1,m] list of data.frames, Input data to GFA in COCOREG format
`prc_th`	[1,1] double, Threshold in percentage of cumulative variance explained PCA components are included until cumulative explained variance reaches prc_th.

Value

A list with elements:

`$data`	[1,m] list of data.frames, Original data reconstructed using only those latent components that are active in all datasets
`$dataid`	string, Dataset identifier string
`$method`	string, Analysis method identifier string
`$wall_time_taken`	[1,1] double, Time taken to run the analysis in seconds

`rename_variables` *Rename variables of a data collection*

Description

Rename variable in all datasets such that the data.frame list conforms to the requirements of CoCoReg.

Usage

```
rename_variables(df_list)
```

Arguments

`df_list`	list of data.frames, The datasets to process

Value

A list of data.frames with changed variable names. Original dimension names are stored as an attribute.

`repmat` *Replicate matrix to create a larger one*

Description

From: http://haky-functions.blogspot.fi/2006/11/repmat-function-matlab.html (accessed 27.3.2015)

Usage

```
repmat(X, m, n)
```

Arguments

`X`	A [I,J] matrix or J element vector, Matrix used as such, vector coerced to a row matrix with dim(X)=[1,J].
`m`	[1,1] integer, Replication count vertically
`n`	[1,1] integer, Replication count horizontally

Value

m*I,n*J	matrix, Replicated data

`RGCCA_cocoreg_interface` *COCOREG style analysis using RGCCA projection*

Description

COCOREG interface used for both input and output.

Usage

```
RGCCA_cocoreg_interface(dflst, tauArr = rep(0.5, length(dflst)))
```

Arguments

`dflst,`	[1,m] list of data.frames, Input data to GFA in COCOREG format
`tauArr,`	[1,m] double, See RGCCA::rgcca() for details

Value

A list with elements:

`$data`	[1,m] list of data.frames, Original data reconstructed using only those latent components that are active in all datasets
`$model`	list, The output RGCCA::rgcca()
`$dataid`	string, Dataset identifier string
`$method`	string, Analysis method identifier string
`$wall_time_taken`	[1,1] double, Time taken to run the analysis in seconds

`rmse` *Compute RMSE between vectors v1 and v2*

Description

Compute RMSE between vectors v1 and v2

Usage

```
rmse(v1, v2, relative = F)
```

Arguments

`v1`	[1,m] numeric, First data vector
`v2`	[1,m] numeric, Second data vector
`relative`	boolean, If TRUE, relate the rmse value to the rmse of v1. If FALSE, just compute RMSE between v1 and v2

Value

1,1 double, RMSE value

`rotation_matrix` *A rotation matrix*

Description

A rotation matrix

Usage

```
rotation_matrix(angle_deg)
```

Arguments

`angle_deg`	[1,1] numeric, Angle in degrees

Value

2,2 matrix, Rotation matrix for making angle_deg 2D rotation

`row_suffle_variability` *Determine the variability of matrices under row shuffling*

Description

Determine the variability of matrices under row shuffling

Usage

```
row_suffle_variability(mat_lst, B = 50)
```

Arguments

`mat_lst`	A list of matrices
`B`	integer, Number of times to sample (shuffle)

Value

B,K matrix, Frobenius norm vectors corresponding to row shuffling

`SCA_cocoreg_interface` *SCA projection using cocoreg interface*

Description

SCA projection using cocoreg interface

Usage

```
SCA_cocoreg_interface(df_list, nfac = 1, type = "sca-p",
  rotation = "none", nstart = 10)
```

Arguments

`df_list,`	[1,m] list of data.frames, Input data to GFA in COCOREG format
`nfac,`	[1,1] int, see multiway::sca() for details
`type,`	string, Type of analysis, see multiway::sca() for details
`rotation,`	string, see multiway::sca() for details
`nstart,`	[1,1] int, see multiway::sca() for details

Value

A list with elements:

`$data`	[1,m] list of data.frames, Original data reconstructed using only those latent components that are active in all datasets
`$model`	list, The output of multiway::sca()
`$dataid`	string, Dataset identifier string
`$method`	string, Analysis method identifier string
`$wall_time_taken`	[1,1] double, Time taken to run the analysis in seconds

`se` *Standard error of mean*

Description

Standard error of mean

Usage

```
se(x, na.rm = T)
```

Arguments

`x`	[1,M] numeric, data vector
`na.rm`	procedure for NA's, passed on to sd(), default: na.rm = T

Value

1,1 numeric, standard error of mean

`ss` *Sum of squares*

Description

Sum of squares

Usage

```
ss(x)
```

Arguments

`x`	[1,m] numeric, A data vector

Value

Sum of squares of x

`svm_sigmoid` *SVM using sigmoid kernel*

Description

SVM using sigmoid kernel

Usage

```
svm_sigmoid(...)
```

Arguments

`...`	Further arguments passed on to e1071::svm()

`to_unit_vec` *Make vector of unit norm*

Description

Make vector of unit norm

Usage

```
to_unit_vec(x)
```

Arguments

`x`	[1,m] double, A vector of data

Value

1,m double, Same vector normalized to unit Euclidean norm

`traverse_nested_list` *Apply fun to the bottom level of a nested list structure*

Description

Used to batch process computation results that are stored into a nested list structure. Analysis results are stored as lists but with class attribute changed. This signals that the recursion into the list structure should end and fun should be applied instead. Can be used e.g. to pick out results from a complex list structure.

Usage

```
traverse_nested_list(lst, fun, exclude_names = NULL, ...)
```

Arguments

`lst`	nested list, A nested list structure to process
`fun`	function object, The function to apply at the bottom level
`exclude_names`	string array, Names of list elements to skip at any level
`...`	Further parameters passed on to fun

Value

A list outputs generated when applying fun to the bottom level of input lst. Bottom level is considered reached when something other than class == 'list' is encountered.

`validate_data` *Validate a data collection for use with cocoreg*

Description

Check that the data collection has all the required properties.

Usage

```
validate_data(df_list)
```

Arguments

`df_list`	list of data.frames, The data collection to validate

Value

A list of data.frames that conform to the requirements

`variability_components` *Compute total, within group and between group variability using fun*

Description

The function used the definition: gvar = tvar - wgvar

Usage

```
variability_components(vec, grp, fun)
```

Arguments

`vec`	[1,M] numeric, Data vector
`grp`	[1,M] integer/character vector, Some grouping of vec
`fun`	function, Function to use when quantifying the variability

Value

A list with elements:

`tvar:`	Total variability
`bgvar:`	Between groups variability, tvar - sum(wgvar_*)
`wgvar_<groupname>:`	Within group variability for each group
`wg_rel:`	sum(wgvar)/tvar
`bg_rel:`	bgvar/tvar

Examples

```
vec <- rnorm(10)
grp <- rep(c("a","b","c"), c(3,3,4))
variability_components(vec, grp, ss)
```

`variation_shared_by` *Return a specific variation component*

Description

Variation can be shared by: 'all' all datasets 'subset' a subset of the datasets (excluding variation already in 'all') 'all_and_subset' a union of the above

The returned data never contains noise (which is considered to be part of each datasets unique variation). The linear toy datasets do not contain variation unique to a dataset other than pure noise.

Usage

```
variation_shared_by(dc, type, center = T, scale = F)
```

Arguments

`dc`	A data collection from one of the create_syn_data_*() functions
`type`	Type of variation to extract, allowed values c('all','subset','all_and_subset')
`center`	(optional) Should the output data be centered?
`scale`	(optional) Should the output data be scaled?

Value

A list of data.frames containing the desired variation component

Examples

```
## Not run:
dc <- create_syn_data_toy()
ldSharedByAll = variation_shared_by(dc, "all", center = F)
ldSharedBySome = variation_shared_by(dc, "subset", center = F)
ldNonUnique = variation_shared_by(dc, "all_and_subset", center = F)
dNoise <- mapply(function(x,y){x-y}, x=dc$data, y=ldNonUnique, SIMPLIFY = F)
ggplot_dclst(list(observed = dc$data,
                shared.by.all = ldSharedByAll,
                shared.by.some = ldSharedBySome,
                noise = dNoise),
            ylim = c(-3, 3))

## End(Not run)
```

var_explained — *Sum-of-squares values showing what portion of variance in dvec is explained by dvec_est*

Description

Computation as in: http://en.wikipedia.org/wiki/Fraction_of_variance_unexplained

ss_est becomes zero if dvec_est equals dvec_0=rep(mean(dvec),length(dvec)). If dvec_est is better estimate than dvec_0, R2 is positive. If dvec_est is worse than dvec_0, R2 is negative.

Usage

```
var_explained(dvec, dvec_est)
```

Arguments

`dvec`	[1,m] numeric, data vector
`dvec_est`	[1,m] numeric, data vector, an estimate of dvec

Value

A list with elements:

`ss_tot:`	Sum of squares in dvec
`ss_est:`	Sum of squares in dvec_est
`ss_err:`	Sum of squares of dvec - dvec_est
`R2:`	Percentage of variance explained i.e. 1 - ss_err/ss_tot

vecnorm — *Compute Euclidean norm of vector*

Description

Convenience function for use with e.g. lapply

Usage

```
vecnorm(x)
```

Arguments

`x`	[1,m] double, A vector of data

Value

1,1 double, Euclidean norm of x

`vector_variability` *Compute "variance" of the vectors var()*

Description

Compute "variance" of the vectors var()

Usage

```
vector_variability(vec_lst, mean_vec = apply(abind::abind(vec_lst, along = 2),
  1, mean))
```

Arguments

`vec_lst`	Data to process as a list of numeric vectors
`mean_vec`	(optional) Desired mean vector as a numeric vector

Value

Variance of data values around mean as a numeric matrix

`wrapper_BGFA` *Run BGFA by Klami et. al. using data format conventions of this repo*

Description

Run BGFA by Klami et. al. using data format conventions of this repo

Usage

```
wrapper_BGFA(df_list, K = 8, Nrep = 2)
```

Arguments

`df_list`	[1,m] list of data.frames, Input data to GFA in COCOREG format
`K`	[1,1] int, The (maximum) number of components; should be high enough to capture all of the components. This can be recognized by at least a few of the components being shut down
`Nrep`	[1,1] int, Number of random initialization used for learning the model

Value

A list, The output of CCAGFA::GFA()

Index