Package 'structToolbox'

October 17, 2020

Type Package

Title Data processing & analysis tools for Metabolomics and other omics

Version 1.0.1

Description An extensive set of data (pre-)processing and analysis methods and tools for metabolomics and other omics, with a strong emphasis on statistics and machine learning. This toolbox allows the user to build extensive and standardised workflows for data analysis. The methods and tools have been implemented using class-based templates provided by the struct (Statistics in R Using Class-based Templates) package. The toolbox includes pre-processing methods (e.g. signal drift and batch correction, normalisation, missing value imputation and scaling), univariate (e.g. ttest, various forms of ANOVA, Kruskal–Wallis test and more) and multivariate statistical methods (e.g. PCA and PLS, including cross-validation and permutation testing) as well as machine learning methods (e.g. Support Vector Machines). The STATistics Ontology (STATO) has been integrated and implemented to provide standardised definitions for the different methods, inputs and outputs.

License GPL-3

Encoding UTF-8

LazyData true

Collate 'AUC_metric_class.R' 'entity_objects.R' 'DFA_class.R' 'anova_class.R' 'HSD_class.R' 'mixed_effect_class.R' 'HSDEM_class.R' 'MTBLS79_dataset_class.R' 'PCA_class.R' 'PCA_plotfcns.R' 'PLSDA_class.R' 'PLSDA_charts.R' 'PLSR_class.R' 'as_data_frame_doc.R' 'autoscale_class.R' 'balanced_accuracy_class.R' 'blank_filter_class.R' 'bootstrap_class.R' 'calculate_doc.R' 'chart_plot_doc.R' 'classical_lsq_class.R' 'confounders_clsq_class.R' 'constant_sum_norm_class.R' 'corr_coef_class.R' 'd_ratio_filter_class.R' 'dataset_chart_classes.R' 'factor_barchart_class.R' 'feature_profile_class.R' 'filter_by_name_class.R' 'filter_na_count.R' 'filter_smeta_class.R' 'fisher_exact_class.R' 'fold_change_class.R' 'fold_change_int_class.R' 'forward_selection_by_rank_class.R' 'ggplot_theme_pub.R' 'glog_class.R' 'grid_search_1d_class.R' 'hca_class.R' 'kfold_xval_class.R' 'kfold_xval_charts.R' 'knn_impute_class.R' 'kw_rank_sum_class.R' 'linear_model_class.R' 'log_transform.R' 'mean_centre_class.R' 'mean_of_medians.R' 'model_apply_doc.R' 'model_predict_doc.R' 'model_reverse_doc.R' 'model_train_doc.R' 'mv_feature_filter_class.R' 'mv_sample_filter_class.R' 'nroot_transform_class.R' 'pairs_filter_class.R' 'paretoscale_class.R' 'permutation_test_class.R' 'permute_sample_order_class.R' 'pqn_norm_method_class.R' 'prop_na_class.R' 'r_squared_class.R' 'rsd_filter.R' 'run_doc.R' 'sb_corr.R' 'split_data_class.R' 'stratified_split_class.R' 'structToolbox.R' 'svm_classifier_class.R' 'tSNE_class.R' 'ttest_class.R' 'vec_norm_class.R' 'wilcox_test_class.R' 'zzz.R'

Depends R (>= 4.0), struct (>= 0.99.10)

Imports ggplot2, ggthemes, grid, gridExtra, methods, scales, sp, stats

RoxygenNote 7.1.0

Suggests agricolae, BiocFileCache, BiocStyle, car, covr, cowplot, e1071, emmeans, ggdendro, knitr, magick, nlme, openxlsx, pls, pmp, reshape2, ropls, rmarkdown, Rtsne, testthat

VignetteBuilder knitr

biocViews WorkflowStep, Metabolomics

git_url https://git.bioconductor.org/packages/structToolbox

git_branch RELEASE_3_11

git_last_commit 8b4c746

git_last_commit_date 2020-05-26

Date/Publication 2020-10-16

Author Gavin Rhys Lloyd [aut, cre], Ralf Johannes Maria Weber [aut]

Maintainer Gavin Rhys Lloyd <`g.r.lloyd@bham.ac.uk`>

R topics documented:

`ANOVA` *ANOVA*

Description

Applies ANOVA to each feature in a DatasetExperiment object.

Usage

```
ANOVA(alpha = 0.05, mtc = "fdr", formula, ss_type = "III", ...)
```

Arguments

`alpha`	The p-value threshold. Default alpha = 0.05.
`mtc`	Multiple test correction method passed to `p.adjust`. Default mtc = 'fdr'.
`formula`	The formula to use for ANOVA. See `lm` for details.
`ss_type`	The type of sum of squares to use. Can be I, II or III. Default is ss_type = 'III'.
`...`	additional slots and values passed to struct_class

Value

ANOVA object

Examples

```
D = iris_DatasetExperiment()
M = ANOVA(formula=y~Species)
M = model_apply(M,D)
```

`as_data_frame` *Convert to data.frame*

Description

convert the outputs of the input model into a data.frame.

Usage

```
## S4 method for signature 'filter_na_count'
as_data_frame(M)

## S4 method for signature 'ttest'
as_data_frame(M)

## S4 method for signature 'wilcox_test'
as_data_frame(M)
```

Arguments

`M`	a model object

Value

A data.frame of model outputs

Examples

```
D = iris_DatasetExperiment()
M = filter_na_count(threshold=50,factor_name='Species')
M= model_apply(M,D)
df = as_data_frame(M)
```

AUC *Area under ROC*

Description

Area under ROC calculated by approximating the curve as a series of trapeziums. Only suitable for 2 classes.

Usage

```
AUC(...)
```

Arguments

`...`	additional slots and values passed to struct_class

Value

A metric object with methods for calculating AUC

struct object

Examples

```
D = iris_DatasetExperiment()
XCV = kfold_xval(folds=5,factor_name='Species') *
      (mean_centre() + PLSDA(number_components=2,factor_name='Species'))
MET = AUC()
XCV = run(XCV,D,MET)
```

autoscale *Autoscale*

Description

Autoscaling centres the columns of the data in a DatasetExperiment object and divides by the standard deviation.

Usage

```
autoscale(mode = "data", ...)
```

Arguments

`mode`	Used to control whether centring is apply to the data, the meta data or both. Can be any one of "data","sample_meta" or "both". default is "data".
`...`	additional slots and values passed to struct_class

Value

A STRUCT model object with methods for autoscaling.

struct object

Examples

```
D = iris_DatasetExperiment()
M = autoscale()
M = model_train(M,D)
M = model_predict(M,D)
```

`balanced_accuracy` *Balanced Accuracy*

Description

Balanced accuracy is the average of the true positive rate and false positive rate.

Usage

```
balanced_accuracy(...)
```

Arguments

`...`	additional slots and values passed to struct_class

Value

A metric object with methods for calculating balanced accuracy.

struct object

Examples

```
D = iris_DatasetExperiment()
XCV = kfold_xval(folds=5,factor_name='Species') *
      (mean_centre() + PLSDA(number_components=2,factor_name='Species'))
MET = balanced_accuracy()
XCV = run(XCV,D,MET)
```

blank_filter *Blank filter*

Description

Filters features based on the features present in blank samples. The median intensity of the samples is compared to the median intensity of the blank samples. Any sample not sufficiently more intense than the blank is removed. This is a wrapper for the blank filter in the PMP package.

Usage

```
blank_filter(
  fold_change = 20,
  blank_label = "blank",
  qc_label = "QC",
  factor_name,
  fraction_in_blank = 0,
  ...
)
```

Arguments

`fold_change`	`numeric(1)`, fold_change minimum fold change between analytical and blank samples.
`blank_label`	`character(1)`, class label used to identify blank samples.
`qc_label`	`character(1)` or `NULL`, class label used to identify QC samples.
`factor_name`	the column name of sample_meta to use
`fraction_in_blank`	`numeric(1)`, value between 0 to 1 to specify fraction in how many blanks peaks should be present.
`...`	additional slots and values passed to struct_class

Value

A struct model object for applying a blank filter

Examples

```
D = iris_DatasetExperiment()
M = blank_filter(fold_change=2,
                 factor_name='Species',
                 blank_label='setosa',
                 qc_label='versicolor')
M = model_apply(M,D)
```

`blank_filter_hist` *plot for blank filter*

Description

plots a histogram of the calculated fold change for the blank filter (median blank / median sample)

Usage

```
blank_filter_hist(...)
```

Arguments

`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
C = blank_filter_hist()
```

`bootstrap` *Bootstrap class*

Description

Applies bootstrapping to a model or model_seq(). Any output from the model can be 'collected' over all bootstrap repetitions for further analysis.

Usage

```
bootstrap(number_of_iterations = 100, collect, ...)
```

Arguments

`number_of_iterations`	The number of bootstrap iterations to run
`collect`	The name of model output to collect over all iterations
`...`	additional slots and values passed to struct_class

Value

A struct iterator object

Examples

```
D = iris_DatasetExperiment()
I = bootstrap(number_of_iterations = 5, collect='vip') *
    (mean_centre() + PLSDA(factor_name = 'Species'))
I = run(I,D,balanced_accuracy())
```

`calculate,AUC-method` *Calculate metric*

Description

Calculate metric

Usage

```
## S4 method for signature 'AUC'
calculate(obj, Y, Yhat)

## S4 method for signature 'balanced_accuracy'
calculate(obj, Y, Yhat)

## S4 method for signature 'r_squared'
calculate(obj, Y, Yhat)
```

Arguments

`obj`	a metric object
`Y`	the true values/group labels
`Yhat`	the predicted values/group labels

Value

a modified metric object

Examples

```
MET = metric()
calculate(MET)
```

`chart_plot,dfa_scores_plot,DFA-method` *chart_plot method*

Description

Plots a chart object

Usage

```
## S4 method for signature 'dfa_scores_plot,DFA'
chart_plot(obj, dobj)

## S4 method for signature 'pca_correlation_plot,PCA'
chart_plot(obj, dobj)
```

```
## S4 method for signature 'pca_scores_plot,PCA'
chart_plot(obj, dobj)

## S4 method for signature 'pca_biplot_plot,PCA'
chart_plot(obj, dobj)

## S4 method for signature 'pca_loadings_plot,PCA'
chart_plot(obj, dobj)

## S4 method for signature 'pca_scree_plot,PCA'
chart_plot(obj, dobj)

## S4 method for signature 'pca_dstat_plot,PCA'
chart_plot(obj, dobj)

## S4 method for signature 'plsda_scores_plot,PLSDA'
chart_plot(obj, dobj)

## S4 method for signature 'plsda_predicted_plot,PLSDA'
chart_plot(obj, dobj)

## S4 method for signature 'plsda_roc_plot,PLSDA'
chart_plot(obj, dobj)

## S4 method for signature 'plsda_vip_plot,PLSDA'
chart_plot(obj, dobj)

## S4 method for signature 'plsda_regcoeff_plot,PLSDA'
chart_plot(obj, dobj)

## S4 method for signature 'plsr_prediction_plot,PLSR'
chart_plot(obj, dobj)

## S4 method for signature 'plsr_residual_hist,PLSR'
chart_plot(obj, dobj)

## S4 method for signature 'plsr_qq_plot,PLSR'
chart_plot(obj, dobj)

## S4 method for signature 'plsr_cook_dist,PLSR'
chart_plot(obj, dobj)

## S4 method for signature 'blank_filter_hist,blank_filter'
chart_plot(obj, dobj)

## S4 method for signature 'confounders_lsq_barchart,confounders_clsq'
chart_plot(obj, dobj)

## S4 method for signature 'confounders_lsq_boxplot,confounders_clsq'
chart_plot(obj, dobj)

## S4 method for signature 'feature_boxplot,DatasetExperiment'
```

```
chart_plot(obj, dobj)

## S4 method for signature 'mv_histogram,DatasetExperiment'
chart_plot(obj, dobj)

## S4 method for signature 'mv_boxplot,DatasetExperiment'
chart_plot(obj, dobj)

## S4 method for signature 'DatasetExperiment_dist,DatasetExperiment'
chart_plot(obj, dobj)

## S4 method for signature 'DatasetExperiment_boxplot,DatasetExperiment'
chart_plot(obj, dobj)

## S4 method for signature 'compare_dist,DatasetExperiment'
chart_plot(obj, dobj, eobj)

## S4 method for signature 'DatasetExperiment_heatmap,DatasetExperiment'
chart_plot(obj, dobj)

## S4 method for signature
## 'DatasetExperiment_factor_barchart,DatasetExperiment'
chart_plot(obj, dobj)

## S4 method for signature 'feature_profile,DatasetExperiment'
chart_plot(obj, dobj)

## S4 method for signature 'fold_change_plot,fold_change'
chart_plot(obj, dobj)

## S4 method for signature 'fs_line,forward_selection_byrank'
chart_plot(obj, dobj)

## S4 method for signature 'glog_opt_plot,glog_transform'
chart_plot(obj, dobj, gobj)

## S4 method for signature 'gs_line,grid_search_1d'
chart_plot(obj, dobj)

## S4 method for signature 'hca_dendrogram,HCA'
chart_plot(obj, dobj)

## S4 method for signature 'kfoldxcv_grid,kfold_xval'
chart_plot(obj, dobj)

## S4 method for signature 'kfoldxcv_metric,kfold_xval'
chart_plot(obj, dobj)

## S4 method for signature 'kw_p_hist,kw_rank_sum'
chart_plot(obj, dobj)

## S4 method for signature 'mv_feature_filter_hist,mv_feature_filter'
```

```
chart_plot(obj, dobj)

## S4 method for signature 'mv_sample_filter_hist,mv_sample_filter'
chart_plot(obj, dobj)

## S4 method for signature 'permutation_test_plot,permutation_test'
chart_plot(obj, dobj)

## S4 method for signature 'pqn_norm_hist,pqn_norm'
chart_plot(obj, dobj)

## S4 method for signature 'rsd_filter_hist,rsd_filter'
chart_plot(obj, dobj)

## S4 method for signature 'svm_plot_2d,SVM'
chart_plot(obj, dobj, gobj)

## S4 method for signature 'tSNE_scatter,tSNE'
chart_plot(obj, dobj)

## S4 method for signature 'wilcox_p_hist,wilcox_test'
chart_plot(obj, dobj)
```

Arguments

obj	a chart object
dobj	a struct object
eobj	a second DatasetExperiment object to compare with the first
gobj	The DatasetExperiment object used with glog_transform

Value

a plot object

Examples

```
C = example_chart()
chart_plot(C,iris_DatasetExperiment())
```

classical_lsq *Classical Least Squares regression*

Description

Classical least squares, where y is the response and X is the design matrix, applied to each feature individually. Here the response is taken from the data matrix and the design matrix is the taken from the specified sample meta data column.

Usage

```
classical_lsq(alpha = 0.05, mtc = "fdr", factor_names, intercept = TRUE, ...)
```

Arguments

`alpha`	p-value threshold for determining significance. Default alpha = 0.05.
`mtc`	multiple test correction method to apply. Can be: holm, hochberg, hommel, bonferroni, BH, BY, fdr or none
`factor_names`	the column name(s) of sample_meta to use
`intercept`	[TRUE] or FALSE to include an intercept term in the fit
`...`	additional slots and values passed to struct_class

Value

A STRUCT method object with functions for applying classical least squares

struct object

Examples

```
D = iris_DatasetExperiment()
M = classical_lsq(factor_names = 'Species')
M = model_apply(M,D)
```

`compare_dist` *Compare distributions*

Description

A combination of plots to compare distributions of samples/features in two datasets

Usage

```
compare_dist(factor_name, ...)
```

Arguments

`factor_name`	the sample_meta colum to use
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D1=MTBLS79_DatasetExperiment(filtered=FALSE)
D2=MTBLS79_DatasetExperiment(filtered=TRUE)
C = compare_dist(factor_name='class')
chart_plot(C,D1,D2)
```

`confounders_clsq` *Check for confounding factors in ttest*

Description

Compares the coefficients for a ttest without including confounding factors to models with confounding factor included. Currently only ttest is supported.

Usage

```
confounders_clsq(
  alpha = 0.05,
  mtc = "fdr",
  factor_name,
  confounding_factors,
  threshold = 0.15,
  ...
)
```

Arguments

`alpha`	p-value threshold for determining significance. Default alpha = 0.05.
`mtc`	multiple test correction method to apply. Can be: holm, hochberg, hommel, bonferroni, BH, BY, fdr or [none]
`factor_name`	the column name of sample_meta to use in regression
`confounding_factors`	the column names of factors potentially confounding with the main factor if interest
`threshold`	the threshold (between 0 and 1) for accepting a factor as confounding
`...`	additional slots and values passed to struct_class

Value

A struct model object with functions for applying classical least squares

struct object

Examples

```
D = MTBLS79_DatasetExperiment()
M = filter_by_name(mode='include',dimension='variable',
        names=colnames(D$data)[1:10]) + # first 10 features
    filter_smeta(mode='exclude',levels='QC',
        factor_name='class') + # reduce to two group comparison
    confounders_clsq(factor_name = 'class',
        confounding_factors=c('sample_order','batch'))
M = model_apply(M,D)
```

confounders_lsq_barchart

barchart of percent change

Description

plots a barchart of the percent change when including a confounding factor in a classical least squares model

Usage

```
confounders_lsq_barchart(feature_to_plot, threshold = 10, ...)
```

Arguments

`feature_to_plot`	the name or index of the feature to be plotted
`threshold`	the threshold to be plotted (in %)
`...`	additional slots and values passed to struct_class

Value

A STRUCT chart object

struct object

Examples

```
D = MTBLS79_DatasetExperiment()
M = filter_by_name(mode='include',dimension='variable',
        names=colnames(D$data)[1:10]) + # first 10 features
    filter_smeta(mode='exclude',levels='QC',
        factor_name='class') + # reduce to two group comparison
    confounders_clsq(factor_name = 'class',
        confounding_factors=c('sample_order','batch'))
M = model_apply(M,D)
C = C=confounders_lsq_barchart(feature_to_plot=1,threshold=15)
chart_plot(C,M[3])
```

confounders_lsq_boxplot

boxplot of percent change

Description

Plots a boxplot of the percent change over all features when including a confounding factor in the ttest

Usage

```
confounders_lsq_boxplot(threshold = 10, ...)
```

Arguments

`threshold`	the threshold to be plotted (in %)
`...`	additional slots and values passed to struct_class

Value

A STRUCT chart object

struct object

Examples

```
D = MTBLS79_DatasetExperiment()
M = filter_by_name(mode='include',dimension='variable',
        names=colnames(D$data)[1:10]) + # first 10 features
    filter_smeta(mode='exclude',levels='QC',
        factor_name='class') + # reduce to two group comparison
    confounders_clsq(factor_name = 'class',
        confounding_factors=c('sample_order','batch'))
M = model_apply(M,D)
C = C=confounders_lsq_boxplot(threshold=15)
chart_plot(C,M[3])
```

`constant_sum_norm` *Normalisation to constant sum*

Description

Applies normalisation to a constant sum, such that the sum of values for each sample after normalisation is equal to 1 (or a scaling factor, if specified).

Usage

```
constant_sum_norm(scaling_factor = 1, ...)
```

Arguments

`scaling_factor`	The sum of all values for a sample after normalisation. Default = 1.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
M = constant_sum_norm()
```

corr_coef *Correlation Coefficient*

Description

Calculates correlation between features and continuous variables.

Usage

```
corr_coef(alpha = 0.05, mtc = "fdr", factor_names, method = "spearman", ...)
```

Arguments

alpha	p-value threshold for determining significance. Default alpha = 0.05.
mtc	multiple test correction method to apply. Can be: holm, hochberg, hommel, bonferroni, BH, BY, fdr or none
factor_names	Sample_meta column names to correlate features with
method	'Calculate "kendall", "pearson" or "spearman" correlation coefficient. Default method = "spearman".'
...	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = MTBLS79_DatasetExperiment(filtered=TRUE)

# subset for this example
D = D[,1:10]

# convert to numeric for this example
D$sample_meta$sample_order=as.numeric(D$sample_meta$sample_order)
D$sample_meta$sample_rep=as.numeric(D$sample_meta$sample_rep)

M = corr_coef(factor_names=c('sample_order','sample_rep'))
M = model_apply(M,D)
```

DatasetExperiment_boxplot *DatasetExperiment boxplot*

Description

Boxplot of values per sample/feature in a DatasetExperiment

Usage

```
DatasetExperiment_boxplot(
  factor_name,
  by_sample = TRUE,
  per_class = TRUE,
  number = 50,
  ...
)
```

Arguments

`factor_name`	the column name of sample_meta to use
`by_sample`	[TRUE] or FALSE to plot by samples or features respectively
`per_class`	[TRUE] or FALSE to plot per level in factro_name
`number`	the number of samples/features to plot
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = MTBLS79_DatasetExperiment()
C = DatasetExperiment_boxplot(factor_name='class',number=10,per_class=FALSE)
chart_plot(C,D)
```

`DatasetExperiment_dist` *Distribution plot*

Description

Visualise distributions of values in features/samples.

Usage

```
DatasetExperiment_dist(factor_name, per_class = TRUE, ...)
```

Arguments

`factor_name`	the sample_meta column to use
`per_class`	= [TRUE] or FALSE to plot distrubutions for each level in factor_name
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = MTBLS79_DatasetExperiment()
C = DatasetExperiment_dist(factor_name='class')
chart_plot(C,D)
```

DatasetExperiment_factor_barchart

DatasetExperiment_factor_barchart class

Description

Bar charts based on groupings by factor. Can plot up to three factors.

Usage

```
DatasetExperiment_factor_barchart(feature_to_plot, factor_names, ...)
```

Arguments

`feature_to_plot`	Column ID of feature to plot.
`factor_names`	Names(s) of factors to plot for a feature
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
C = DatasetExperiment_factor_barchart(factor_names='Species',feature_to_plot='Petal.Width')
chart_plot(C,D)
```

DatasetExperiment_heatmap

DatasetExperiment_heatmap class

Description

plots a DatasetExperiment as a heatmap

Usage

```
DatasetExperiment_heatmap(na_colour = "#FF00E4", ...)
```

Arguments

`na_colour`	A hex colour code to use for missing values
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
C = DatasetExperiment_heatmap()
```

DFA — *Discriminant Factor Analysis (DFA)*

Description

Applies Discriminant Factor Analysis to a dataset.

Usage

```
DFA(factor_name, number_components = 2, ...)
```

Arguments

`factor_name`	The sample_meta column name containing group labels
`number_components`	The number of discriminant factors to calculate
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
M = DFA(factor_name='Species')
M = model_apply(M,D)
```

dfa_scores_plot *dfa_scores_plot class*

Description

2d scatter plot of discriminant factor scores.

Usage

```
dfa_scores_plot(
  components = c(1, 2),
  points_to_label = "none",
  factor_name,
  ellipse = "all",
  label_filter = character(0),
  label_factor = "rownames",
  label_size = 3.88,
  ...
)
```

Arguments

`components`	The discriminant factors to plot (`numeric(2)`)
`points_to_label`	"none", "all", or "outliers" will be labelled on the plot.
`factor_name`	The sample_meta column name to use for colouring the points. You can provide up to two factors for this plot.
`ellipse`	"all" will plot all ellipses, "group" will only plot group ellipses, "none" will not plot any ellipses and "sample" will plot ellipse for all samples (ignoring group).
`label_filter`	Only include labels for samples in the group specified by label_filter. If zero length then all labels will be included.
`label_factor`	The sample_meta column to use for labelling the samples. If 'rownames' then the rownames will be used.
`label_size`	The text size of the labels.NB ggplot units, not font size units. Default 3.88.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
M = mean_centre() + DFA(factor_name='Species')
M = model_apply(M,D)
C = dfa_scores_plot(factor_name = 'Species')
chart_plot(C,M[2])
```

`dratio_filter` *D ratio filter*

Description

Filters features based on their D ratio, which is the ratio of technical to sample variance.

Usage

```
dratio_filter(threshold = 20, qc_label = "QC", factor_name, ...)
```

Arguments

`threshold`	D ratio threshold. Features with a d-ratio larger than this value are removed.
`qc_label`	the label used to identify QC labels
`factor_name`	the the sample_meta data column containing the QC labels
`...`	additional slots and values passed to struct_class

Value

A struct method object with functions for filtering using the d-ratio.

Examples

```
D = MTBLS79_DatasetExperiment()
M = dratio_filter(threshold=20,qc_label='QC',factor_name='class')
M = model_apply(M,D)
```

`feature_boxplot` *Feature boxplots*

Description

Plots a boxplot of a chosen feature for each group of a input factor.

Usage

```
feature_boxplot(
  label_outliers = TRUE,
  feature_to_plot,
  factor_name,
  show_counts = TRUE,
  ...
)
```

Arguments

`label_outliers`	[TRUE] or FALSE to label outliers on the plot
`feature_to_plot`	the column id to plot.
`factor_name`	the sample_meta column to use
`show_counts`	[TRUE] or FALSE to include the number of samples on the plot
`...`	additional slots and values passed to struct_class

Value

A struct chart object

Examples

```
D = MTBLS79_DatasetExperiment
C = feature_boxplot(factor_name='Species',feature_to_plot='Petal.Width')
chart_plot(C,D)
```

`feature_profile` *Feature profile class*

Description

Scatter plot of a feature against measurement order with limits for samples and quality control samples.

Usage

```
feature_profile(
  run_order,
  qc_label,
  qc_column,
  colour_by,
  feature_to_plot,
  ...
)
```

Arguments

`run_order`	the sample_meta column containing the measurement order of the samples
`qc_label`	the label used to identify QC samples
`qc_column`	the sample_meta column containing the QC labels
`colour_by`	the sample_meta column to use to colour the plot
`feature_to_plot`	the column id of the feature to plot
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = MTBLS79_DatasetExperiment()
C = feature_profile(run_order='sample_order',
    qc_label='QC',
    qc_column='class',
    colour_by='class',
    feature_to_plot=1)
chart_plot(C,D)
```

---

filter_by_name *Filter by name*

Description

A filter to subsample a DatasetExperiment object based on sample or feature name, id, row/column index or using a vector of TRUE/FALSE.

Usage

```
filter_by_name(mode = "exclude", dimension = "sample", names, ...)
```

Arguments

`mode`	"include" or ["exclude"] to subsample a DatasetExperiment by including or excluding samples/features based on the provided labels
`dimension`	["sample"] or "variable" to filter by sample or feature labels
`names`	the sample/feature identifiers to filter by. Can provide column names, column indices or logical.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = MTBLS79_DatasetExperiment()
M = filter_by_name(mode='exclude',dimension='variable',names=c(1,2,3))
M = model_apply(M,D)
```

`filter_na_count` *filter_na_count class*

Description

Filters features by the number of NA per class

Usage

```
filter_na_count(threshold, factor_name, ...)
```

Arguments

`threshold`	the maximum number of NA allowed per level of factor_name
`factor_name`	the sample_meta column name to use
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = MTBLS79_DatasetExperiment()
M = filter_na_count(threshold=3,factor_name='class')
M = model_apply(M,D)
```

`filter_smeta` *filter_smeta class*

Description

A filter to subset a DatasetExperiment object based on sample meta data.

Usage

```
filter_smeta(mode = "include", levels, factor_name, ...)
```

Arguments

`mode`	= ['include'] or 'exclude' to include or exclude samples based on the provided labels
`levels`	a list of level names to include/exclude
`factor_name`	the sample_meta column name to use
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = MTBLS79_DatasetExperiment()
M = filter_smeta(mode='exclude',levels='QC',factor_name='QC')
M = model_apply(M,D)
```

---

fisher_exact *fisher_exact class*

Description

Fisher's exact test (FET). Applies FET for all features in a DatasetExperiment.

Usage

```
fisher_exact(alpha = 0.05, mtc = "fdr", factor_name, factor_pred, ...)
```

Arguments

`alpha`	the p-value threshold to declare a result 'significant'
`mtc`	multiple test correction method
`factor_name`	the sample_meta column to use
`factor_pred`	A data.frame, with a factor of predicted group labels to compare with factor_name. Can be a data frame with a factor of predictions for each feature.'
`...`	additional slots and values passed to struct_class

Value

A struct model with functions for applying fisher exact test.

Examples

```
# load some data
D=MTBLS79_DatasetExperiment()

# prepare predictions based on NA
pred=as.data.frame(is.na(D$data))
pred=lapply(pred,factor,levels=c(TRUE,FALSE))
pred=as.data.frame(pred)

# apply method
M = fisher_exact(alpha=0.05,mtc='fdr',factor_name='class',factor_pred=pred)
M=model_apply(M,D)
```

`fold_change` *fold change class*

Description

Calculates fold change between groups for all features in a DatasetExperiment, based on a log transform and t-test.

Usage

```
fold_change(
  alpha = 0.05,
  factor_name,
  paired = FALSE,
  sample_name = character(0),
  threshold = 2,
  control_group = character(0),
  ...
)
```

Arguments

`alpha`	confidence level to use for intervals
`factor_name`	the sample_meta column to use
`paired`	TRUE or [FALSE] to account for paired samples
`sample_name`	the sample_meta column name to use for a paired samples
`threshold`	a threshold to define fold change as 'significant'.
`control_group`	a level of factor name to use as the control group for calculations.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = MTBLS79_DatasetExperiment()
M = fold_change(factor_name='class')
M = model_apply(M,D)
```

fold_change_int — *fold change for interactions class*

Description

Calculates fold change between groups for interactions between levels of factors. Note that paired forced to FALSE for all comparisons.

Usage

```
fold_change_int(
  alpha = 0.05,
  factor_name,
  threshold = 2,
  control_group = character(0),
  ...
)
```

Arguments

alpha	confidence level to use for intervals
factor_name	the sample_meta column to use
threshold	a threshold to define fold change as 'significant'.
control_group	a level of factor name to use as the control group for calculations.
...	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = MTBLS79_DatasetExperiment()
D=D[,1:10,drop=FALSE]
M = filter_smeta(mode='exclude',levels='QC',factor_name='class') +
    fold_change_int(factor_name=c('class','batch'))
M = model_apply(M,D)
```

fold_change_plot — *fold_change plot*

Description

Plots fold change with error bars for a limited number of features.

Usage

```
fold_change_plot(number_features = 20, orientation = "portrait", ...)
```

Arguments

`number_features`	The number of features to display on the plot
`orientation`	The orientation of the plot (portrait or landscape). Portrait is default.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
C = fold_change_plot()
```

---

forward_selection_byrank

forward selection by rank

Description

Forward selection by rank is a stepwise procedures that includes features incrementally based on their rank. Any measure for ranking the features may be used e.g. PLS VIP score, ttest p-value etc.

Usage

```
forward_selection_byrank(
  min_no_vars = 1,
  max_no_vars = 100,
  step_size = 1,
  factor_name,
  variable_rank,
  ...
)
```

Arguments

`min_no_vars`	minimum number of features to test
`max_no_vars`	maximum numbe ro features to test
`step_size`	the size of the incremenent between min and max no of vars
`factor_name`	the sample-meta colum to use
`variable_rank`	a vector of values that can be used to rank the features, where the smallest value is the first rank.
`...`	additional slots and values passed to struct_class

Value

A struct object

Examples

```
# some data
D = MTBLS79_DatasetExperiment(filtered=TRUE)

# normalise, impute and scale then remove QCs
P = pqn_norm(qc_label='QC',factor_name='class') +
    knn_impute(neighbours=5) +
    glog_transform(qc_label='QC',factor_name='class') +
    filter_smeta(mode='exclude',levels='QC',factor_name='class')
P = model_apply(P,D)
D = predicted(P)

# forward selection using a PLSDA model
M = forward_selection_byrank(factor_name='class',
                             min_no_vars=2,
                             max_no_vars=11,
                             variable_rank=1:2063) *
    (mean_centre() + PLSDA(number_components=1,
                           factor_name='class'))
M = run(M,D,balanced_accuracy())
```

`fs_line` *forward_selection_plot*

Description

Plots the result of the evaluated models against the values the number of features within the search range for forward_selection_by_rank objects.

Usage

```
fs_line(...)
```

Arguments

`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
# some data
D = MTBLS79_DatasetExperiment(filtered=TRUE)

# normalise, impute and scale then remove QCs
P = pqn_norm(qc_label='QC',factor_name='class') +
    knn_impute(neighbours=5) +
    glog_transform(qc_label='QC',factor_name='class') +
    filter_smeta(mode='exclude',levels='QC',factor_name='class')
P = model_apply(P,D)
```

```
D = predicted(P)

# forward selection using a PLSDA model
M = forward_selection_byrank(factor_name='class',
                             min_no_vars=2,
                             max_no_vars=11,
                             variable_rank=1:2063) *
    (mean_centre() + PLSDA(number_components=1,
                           factor_name='class'))
M = run(M,D,balanced_accuracy())

# chart
C = fs_line()
chart_plot(C,M)
```

glog_opt_plot *glog transform optimisation plot*

Description

plots the SSE error vs lambda for glog transform

Usage

```
glog_opt_plot(plot_grid = 100, ...)
```

Arguments

plot_grid	the resolution of the search space for plotting
...	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
M = glog_transform(qc_label='versicolor',factor_name='Species')
M = model_apply(M,D)
C = glog_opt_plot()
chart_plot(C,M,D)
```

`glog_transform` *glog transform*

Description

applies a glog transform to the input data

Usage

```
glog_transform(qc_label = "QC", factor_name, ...)
```

Arguments

`qc_label`	The label used to identify QC samples
`factor_name`	The sample_meta column name containing QC labels
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
M = glog_transform(qc_label='versicolor',factor_name='Species')
M = model_apply(M,D)
```

`grid_search_1d` *grid_search_1d class*

Description

Carries out a grid search for a single parameter to try and identify the ’best’ value for the parameter based on the input metric.

Usage

```
grid_search_1d(
  param_to_optimise,
  search_values,
  model_index,
  factor_name,
  max_min = "min",
  ...
)
```

Arguments

`param_to_optimise`	The name of an input parameter of the model the optimise
`search_values`	A vector of values to search for the optimum
`model_index`	A number indicating which step of a model_seq is to be optimised
`factor_name`	The sample_meta column name to use
`max_min`	'A string 'max' or 'min' to indicate whether to maximise or minimise the metric
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = MTBLS79_DatasetExperiment()
# some preprocessing
M = pqn_norm(qc_label='QC',factor_name='class') +
    knn_impute() +
    glog_transform(qc_label='QC',factor_name='class') +
    filter_smeta(factor_name='class',levels='QC',mode='exclude')
M=model_apply(M,D)
D=predicted(M)

# reduce number of features for this example
D=D[,1:10]

# optmise number of components for PLS model
I = grid_search_1d(param_to_optimise='number_components',search_values=1:5,
        model_index=2,factor_name='class') *
        (mean_centre()+PLSDA(factor_name='class'))
I = run(I,D,balanced_accuracy())
```

`gs_line` *grid_search_plot*

Description

plots the result of the evaluated models for against the values of the optimisation paramter within the search range.

Usage

```
gs_line(...)
```

Arguments

`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
C = gs_line()
```

HCA *HCA method class*

Description

HCA method class. Calculate a hierarchical clustering for the input data.

Usage

```
HCA(
  dist_method = "euclidean",
  cluster_method = "complete",
  minkowski_power = 2,
  factor_name,
  ...
)
```

Arguments

`dist_method`	The distance method to use for clustering. Can be any one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski". Default is "euclidean".
`cluster_method`	The clustering method to use. Can be any one of "ward.D", "ward.D2", "single", "complete", "average", "mcquitty", "median" or "centroid". Default is 'complete'.
`minkowski_power`	This parameter is only used when `dist_method = 'minkowski'`.
`factor_name`	The sample_meta column to use.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
M = HCA(factor_name='Species')
M = model_apply(M,D)
```

`hca_dendrogram` *hca_dendrogram class*

Description

plots a dendrogram for HCA

Usage

```
hca_dendrogram(...)
```

Arguments

`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
C = hca_dendrogram()
```

`HSD` *HSD model class*

Description

Tukey's honest significant difference. Usually used in conjunction with ANOVA, this model compares classes in a pairwise fashion to try to identify which groups are different to the others (if any).

Usage

```
HSD(alpha = 0.05, mtc = "fdr", formula, unbalanced = FALSE, ...)
```

Arguments

`alpha`	The p-value threshold. Default alpha = 0.05.
`mtc`	Multiple test correction method passed to `p.adjust`. Default mtc = 'fdr'.
`formula`	The formula to use. See `lm` for details.
`unbalanced`	TRUE or FALSE to apply correction for unbalanced designs. Default is FALSE.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
M = HSD(formula=y~Species)
M = model_apply(M,D)
```

HSDEM *HSD model class using estimated marginal means*

Description

HSD model class using estimate marginal means, for use with mixed effects designs.

Usage

```
HSDEM(alpha = 0.05, mtc = "fdr", formula, ...)
```

Arguments

alpha	The p-value threshold. Default alpha = 0.05.
mtc	Multiple test correction method passed to `p.adjust`. Default mtc = 'fdr'.
formula	The formula to use. See `lm` for details.
...	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
D$sample_meta$id=rownames(D) # dummy id column
M = HSDEM(formula = y~Species+ Error(id/Species))
M = model_apply(M,D)
```

---

kfoldxcv_grid *kfoldxcv_grid class*

Description

Plot of cross validation predictions vs the true values.

Usage

```
kfoldxcv_grid(factor_name, level, ...)
```

Arguments

`factor_name`	The sample_meta column name to use.
`level`	The level of the factor to plot
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
I = kfold_xval(factor_name='Species') *
    (mean_centre() + PLSDA(factor_name='Species'))
I = run(I,D,balanced_accuracy())

C = kfoldxcv_grid(factor_name='Species',level='setosa')
chart_plot(C,I)
```

`kfoldxcv_metric` *kfoldxcv_metric class*

Description

A box plot of the calculated cross validation metric over all iterations.

Usage

```
kfoldxcv_metric(...)
```

Arguments

`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
C = kfoldxcv_metric()
```

kfold_xval *kfold_xval model class*

Description

Applies k-fold crossvalidation to a model or model_seq()

Usage

```
kfold_xval(folds = 10, method = "venetian", factor_name, ...)
```

Arguments

folds	The number of cross-validation folds
method	The method for selecting samples in each fold. Can be one of "venetian", "blocks" or "random". Default is "venetian".
factor_name	The sample_meta column name to use.
...	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
I = kfold_xval(factor_name='Species') *
    (mean_centre() + PLSDA(factor_name='Species'))
I = run(I,D,balanced_accuracy())
```

knn_impute *knn missing value imputation*

Description

Applies a k-nearest neighbour approach to impute missing values.

Usage

```
knn_impute(
  neighbours = 5,
  sample_max = 50,
  feature_max = 50,
  by = "features",
  ...
)
```

Arguments

`neighbours`	The number of neighbours to use for imputation.
`sample_max`	Maximum percentage of missing values in any sample. Default = 50.
`feature_max`	Maximum percentage of missing values in any feature. Default = 50.
`by`	Impute by similar "samples" or "features". Default = "features".
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
M = knn_impute()
```

kw_p_hist *plot histogram of p values*

Description

plots a histogram of p values

Usage

```
kw_p_hist(...)
```

Arguments

`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
C = kw_p_hist()
```

`kw_rank_sum` *kruskal-wallis model class*

Description

Calculate kw-test for all features in a DatasetExperiment. A non-parametric 1-way ANOVA.

Usage

```
kw_rank_sum(alpha = 0.05, mtc = "fdr", factor_names, ...)
```

Arguments

`alpha`	The p-value threshold. Default alpha = 0.05.
`mtc`	Multiple test correction method passed to `p.adjust`. Default mtc = 'fdr'.
`factor_names`	The sample_meta column name to use
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
M = kw_rank_sum(factor_names='Species')
M = model_apply(M,D)
```

`linear_model` *linear model class*

Description

Wrapper for R lm. See `lm` for details.

Usage

```
linear_model(formula, na_action = "na.omit", contrasts = list(), ...)
```

Arguments

`formula`	The formula to use.
`na_action`	The action to take when missing values are present. Can any one of 'na.omit','na.fail','na.exclude' or 'na.pass'. Default is 'na.omit'.
`contrasts`	The contrasts for this model. If zero length then the default contrasts are used.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
M = linear_model(formula = y~Species)
```

`log_transform` *log transform*

Description

Applies a log transform to the input data

Usage

```
log_transform(base = 10, ...)
```

Arguments

`base`	The base of the logarithm. Default is 10, resulting in a log10 transformation of the data.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
M = log_transform()
```

`mean_centre` *mean_centre model class*

Description

Mean centres the columns of a DatasetExperiment object. Can also centre the meta data for e.g regression models, if required.

Usage

```
mean_centre(mode = "data", ...)
```

Arguments

`mode`	Used to control whether centring is apply to the data, the meta data or both. Can be any one of "data","sample_meta" or "both". default is "data".
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
M = mean_centre()
```

`mean_of_medians` *Mean of median adjustment*

Description

Applies an offset to the data such that the mean of the medians is equal for all samples.

Usage

```
mean_of_medians(factor_name, ...)
```

Arguments

`factor_name`	the column sample of sample_meta to use. Mean of medians will be applied based on the levels in this factor.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
M = mean_of_medians(factor_name='Species')
M = model_apply(M,D)
```

`mixed_effect` *Mixed Effects model class*

Description

Mixed Effects model class. Applies RE model for all features in a DatasetExperiment

Usage

```
mixed_effect(alpha = 0.05, mtc = "fdr", formula, ss_type = "marginal", ...)
```

Arguments

`alpha`	The p-value threshold. Default alpha = 0.05.
`mtc`	Multiple test correction method passed to `p.adjust`. Default mtc = 'fdr'.
`formula`	The formula to use. See `aov` for details.
`ss_type`	Type of sum of squares to use. "marginal" = Type III sum of squares, and "sequential" = Type II. Default is "marginal".
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
D$sample_meta$id=rownames(D) # dummy id column
M = mixed_effect(formula = y~Species+ Error(id/Species))
M = model_apply(M,D)
```

model_apply,ANOVA,DatasetExperiment-method

Apply method

Description

Applies method to the input DatasetExperiment

Usage

```
## S4 method for signature 'ANOVA,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'HSD,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'mixed_effect,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'HSDEM,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'classical_lsq,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'confounders_clsq,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'constant_sum_norm,DatasetExperiment'
model_apply(M, D)
```

```
## S4 method for signature 'corr_coef,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'filter_by_name,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'filter_smeta,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'fisher_exact,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'fold_change,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'fold_change_int,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'HCA,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'knn_impute,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'kw_rank_sum,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'log_transform,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'mean_of_medians,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'mv_sample_filter,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'nroot_transform,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'pairs_filter,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'pqn_norm,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'prop_na,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'rsd_filter,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'sb_corr,DatasetExperiment'
```

```
model_apply(M, D)

## S4 method for signature 'split_data,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'stratified_split,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'tSNE,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'ttest,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'vec_norm,DatasetExperiment'
model_apply(M, D)

## S4 method for signature 'wilcox_test,DatasetExperiment'
model_apply(M, D)
```

Arguments

`M`	a method object
`D`	another object used by the first

Value

Returns a modified method object

Examples

```
M=model()
model_apply(M,DatasetExperiment())
```

`model_predict,DFA,DatasetExperiment-method`
Model prediction

Description

Apply a model using the input DatasetExperiment. Assumes the model is trained first.

Usage

```
## S4 method for signature 'DFA,DatasetExperiment'
model_predict(M, D)

## S4 method for signature 'PCA,DatasetExperiment'
model_predict(M, D)

## S4 method for signature 'PLSDA,DatasetExperiment'
```

```
model_predict(M, D)

## S4 method for signature 'PLSR,DatasetExperiment'
model_predict(M, D)

## S4 method for signature 'autoscale,DatasetExperiment'
model_predict(M, D)

## S4 method for signature 'blank_filter,DatasetExperiment'
model_predict(M, D)

## S4 method for signature 'constant_sum_norm,DatasetExperiment'
model_predict(M, D)

## S4 method for signature 'dratio_filter,DatasetExperiment'
model_predict(M, D)

## S4 method for signature 'filter_by_name,DatasetExperiment'
model_predict(M, D)

## S4 method for signature 'filter_na_count,DatasetExperiment'
model_predict(M, D)

## S4 method for signature 'filter_smeta,DatasetExperiment'
model_predict(M, D)

## S4 method for signature 'glog_transform,DatasetExperiment'
model_predict(M, D)

## S4 method for signature 'linear_model,DatasetExperiment'
model_predict(M, D)

## S4 method for signature 'mean_centre,DatasetExperiment'
model_predict(M, D)

## S4 method for signature 'mv_feature_filter,DatasetExperiment'
model_predict(M, D)

## S4 method for signature 'mv_sample_filter,DatasetExperiment'
model_predict(M, D)

## S4 method for signature 'pareto_scale,DatasetExperiment'
model_predict(M, D)

## S4 method for signature 'SVM,DatasetExperiment'
model_predict(M, D)

## S4 method for signature 'vec_norm,DatasetExperiment'
model_predict(M, D)
```

Arguments

M	a model object

D	a DatasetExperiment object

Value

Returns a modified model object

Examples

```
M = example_model()
M = model_predict(M,iris_DatasetExperiment())
```

`model_reverse,autoscale,DatasetExperiment-method`
Reverse preprocessing

Description

Reverse the effect of a preprocessing step on a DatasetExperiment.

Usage

```
## S4 method for signature 'autoscale,DatasetExperiment'
model_reverse(M, D)

## S4 method for signature 'mean_centre,DatasetExperiment'
model_reverse(M, D)
```

Arguments

M	a model object
D	a DatasetExperiment object

Value

Returns a modified DatasetExperiment object

Examples

```
M = example_model()
D = model_reverse(M,iris_DatasetExperiment())
```

`model_train,DFA,DatasetExperiment-method`

Train a model

Description

Trains a model using the input DatasetExperiment

Usage

```
## S4 method for signature 'DFA,DatasetExperiment'
model_train(M, D)

## S4 method for signature 'PCA,DatasetExperiment'
model_train(M, D)

## S4 method for signature 'PLSDA,DatasetExperiment'
model_train(M, D)

## S4 method for signature 'PLSR,DatasetExperiment'
model_train(M, D)

## S4 method for signature 'autoscale,DatasetExperiment'
model_train(M, D)

## S4 method for signature 'blank_filter,DatasetExperiment'
model_train(M, D)

## S4 method for signature 'constant_sum_norm,DatasetExperiment'
model_train(M, D)

## S4 method for signature 'dratio_filter,DatasetExperiment'
model_train(M, D)

## S4 method for signature 'filter_by_name,DatasetExperiment'
model_train(M, D)

## S4 method for signature 'filter_na_count,DatasetExperiment'
model_train(M, D)

## S4 method for signature 'filter_smeta,DatasetExperiment'
model_train(M, D)

## S4 method for signature 'glog_transform,DatasetExperiment'
model_train(M, D)

## S4 method for signature 'linear_model,DatasetExperiment'
model_train(M, D)

## S4 method for signature 'mean_centre,DatasetExperiment'
model_train(M, D)
```

```
## S4 method for signature 'mv_feature_filter,DatasetExperiment'
model_train(M, D)

## S4 method for signature 'mv_sample_filter,DatasetExperiment'
model_train(M, D)

## S4 method for signature 'pareto_scale,DatasetExperiment'
model_train(M, D)

## S4 method for signature 'SVM,DatasetExperiment'
model_train(M, D)

## S4 method for signature 'vec_norm,DatasetExperiment'
model_train(M, D)
```

Arguments

`M`	a model object
`D`	a DatasetExperiment object

Value

Returns a modified model object

Examples

```
M = example_model()
M = model_train(M,iris_DatasetExperiment())
```

`MTBLS79_DatasetExperiment` *MTBLS79: Direct infusion mass spectrometry metabolomics dataset: a benchmark for data processing and quality control*

Description

Direct-infusion mass spectrometry (DIMS) metabolomics is an important approach for characterising molecular responses of organisms to disease, drugs and the environment. Increasingly large-scale metabolomics studies are being conducted, necessitating improvements in both bioanalytical and computational workflows to maintain data quality. This dataset represents a systematic evaluation of the reproducibility of a multi-batch DIMS metabolomics study of cardiac tissue extracts. It comprises of twenty biological samples (cow vs. sheep) that were analysed repeatedly, in 8 batches across 7 days, together with a concurrent set of quality control (QC) samples. Data are presented from each step of the workflow and are available in MetaboLights (https://www.ebi.ac.uk/metabolights/MTBLS79)

Usage

```
MTBLS79_DatasetExperiment(filtered = FALSE)
```

Arguments

`filtered`	TRUE to load data with quality control filters already applied, or FALSE to load the unfiltered data. Default is FALSE. The raw data is available from (https://www.ebi.ac.uk/metabolights/MTBLS79) and as an R dataset in the `pmp` package, available on Bioconductor.

Value

DatasetExperiment object

Examples

```
D = MTBLS79_DatasetExperiment()
summary(D)
```

`mv_boxplot` *mv_boxplot class*

Description

Boxplot of the numbers of missing values per sample/feature

Usage

```
mv_boxplot(
  label_outliers = TRUE,
  by_sample = TRUE,
  factor_name,
  show_counts = TRUE,
  ...
)
```

Arguments

`label_outliers`	TRUE or FALSE to label outliers on the plot.
`by_sample`	TRUE to plot missing values by sample, or FALSE to plot for features.
`factor_name`	The sample_meta column to use.
`show_counts`	TRUE to show a count of the number of items used to create the boxplot on the chart.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = MTBLS79_DatasetExperiment()
C = mv_boxplot(factor_name='class')
chart_plot(C,D)
```

`mv_feature_filter` *filter features by fraction of missing values*

Description

Filters features by the percent number of missing values, based on class labels if required.

Usage

```
mv_feature_filter(
  threshold = 20,
  qc_label = "QC",
  method = "QC",
  factor_name,
  ...
)
```

Arguments

`threshold`	The max percentage missing values in a feature, above which the feature is removed. Default is 20.
`qc_label`	The label of the QC samples in the named sample_meta column.
`method`	"within_all" applies filter within classes,"within_one" applies filter within any one class, "QC" applies filter within QC samples, "across" applies filter ignoring class.
`factor_name`	The name of the sample_meta column to use.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
M = mv_feature_filter(factor_name='Species',qc_label='versicolor')
M = model_apply(M,D)
```

`mv_feature_filter_hist` *plot for missing value sample filter*

Description

plots a histogram of

Usage

```
mv_feature_filter_hist(...)
```

Arguments

`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
C = mv_feature_filter_hist()
```

`mv_histogram` *mv_histogram class*

Description

histograms indicating the numbers of missing values per sample/feature

Usage

```
mv_histogram(label_outliers = TRUE, by_sample = TRUE, ...)
```

Arguments

`label_outliers`	[TRUE] or FALSE to label outliers on the plot
`by_sample`	[TRUE] to plot by sample or FALSE to plot by features
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = MTBLS79_DatasetExperiment()
C = mv_histogram(label_outliers=FALSE,by_sample=FALSE)
chart_plot(C,D)
```

`mv_sample_filter` *Missing value filter (samples)*

Description

Filters samples based on the percent number of missing values.

Usage

```
mv_sample_filter(mv_threshold = 20, ...)
```

Arguments

`mv_threshold`	The max percentage of missing values, above which the sample is removed.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
C = mv_sample_filter()
```

`mv_sample_filter_hist` *plot for missing value sample filter*

Description

plots a histogram of

Usage

```
mv_sample_filter_hist(...)
```

Arguments

`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
C = mv_sample_filter_hist()
```

`nroot_transform` *nroot transform*

Description

Applies an nth root transform to the data

Usage

```
nroot_transform(root = 2, ...)
```

Arguments

`root`	The nth root of the transform. Default is 2, resulting in a square-root transformation of the data.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
M = nroot_transform()
```

`pairs_filter` *Pairs filter*

Description

Filters samples for paired analysis, ensuring each sample id is present in all groups.

Usage

```
pairs_filter(factor_name, sample_id, ...)
```

Arguments

`factor_name`	(character) the column name of sample_meta containing the labels
`sample_id`	(character) the column name of sample_meta containing the sample ids
`...`	additional slots and values passed to struct_class

Value

A STRUCT method object with functions for applying a pairs filter

struct object

Examples

```
M=pairs_filter(factor_name='Class',sample_id='ids')
```

`pareto_scale` *Pareto scaling*

Description

Pareto scaling centres the columns of the data in a DatasetExperiment object and divides by the square root of the standard deviation.

Usage

```
pareto_scale(...)
```

Arguments

`...`	additional slots and values passed to struct_class

Value

A STRUCT model object with methods for pareto scaling.

struct object

Examples

```
D = iris_DatasetExperiment()
M = pareto_scale()
M = model_train(M,D)
M = model_predict(M,D)
```

`PCA` *PCA model class*

Description

Principal Component Analysis (PCA) model class. This object can be used to train/apply PCA mdoels to DatasetExperiment objects.

Usage

```
PCA(number_components = 2, ...)
```

Arguments

`number_components`	The number of principal components to retain
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
M = PCA()
```

`pca_biplot_plot` *pca_biplot_plot class*

Description

2d scatter plot of principal component scores overlaid with principal component loadings.

Usage

```
pca_biplot_plot(
  components = c(1, 2),
  points_to_label = "none",
  factor_name,
  scale_factor = 0.95,
  style = "points",
  label_features = FALSE,
  ...
)
```

Arguments

`components`	The principal components to plot (`numeric(2)`)
`points_to_label`	"none", "all", or "outliers" will be labelled on the plot.
`factor_name`	The sample_meta column name to use for colouring the points. You can provide up to two factors for this plot.
`scale_factor`	Scaling factor to apply to loadings. Default = 0.95.
`style`	Plot style for loadings. Can be 'points' (default) or 'arrows'.
`label_features`	'Include feature labels from this variable meta column. Special keyword "rownames" will use the rownames of the variable_meta data.frame'
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
C = pca_biplot_plot(factor_name='Species')
```

pca_correlation_plot *pca_correlation_plot class*

Description

Plots the correlation between features and selected components.

Usage

```
pca_correlation_plot(components = c(1, 2), ...)
```

Arguments

`components`	The principal components to plot (`numeric(2)`)
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
C = pca_correlation_plot()
```

pca_dstat_plot *pca_dstat_plot_plot class*

Description

Bar chart showing mahalanobis distance from the mean in PCA scores space. A threshold is plotted at a chosen confidence as an indicator for rejecting outliers.

Usage

```
pca_dstat_plot(number_components = 2, alpha = 0.05, ...)
```

Arguments

`number_components`	The number of components to use.
`alpha`	The confidence level to plot.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
C = pca_dstat_plot()
```

pca_loadings_plot — *pca_loadings_plot class*

Description

2d scatter plot of princpal component loadings.

Usage

```
pca_loadings_plot(
  components = c(1, 2),
  style = "points",
  label_features = NULL,
  ...
)
```

Arguments

`components`	The principal components to plot (`numeric(2)`)
`style`	Plot style for loadings. Can be 'points' (default) or 'arrows'.
`label_features`	'A list of labels to use, one for each feature. Special keyword "rownames" will use the rownames of the variable_meta data.frame'
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
C = pca_loadings_plot()
```

pca_scores_plot — *pca_scores_plot class*

Description

2d scatter plot of principal component scores.

Usage

```
pca_scores_plot(
  components = c(1, 2),
  points_to_label = "none",
  factor_name,
  ellipse = "all",
  label_filter = character(0),
  label_factor = "rownames",
  label_size = 3.88,
  ...
)
```

Arguments

`components`	The principal components to plot (`numeric(2)`)
`points_to_label`	"none", "all", or "outliers" will be labelled on the plot.
`factor_name`	The sample_meta column name to use for colouring the points. You can provide up to two factors for this plot.
`ellipse`	"all" will plot all ellipses, "group" will only plot group ellipses, "none" will not plot any ellipses and "sample" will plot ellipse for all samples (ignoring group).
`label_filter`	Only include labels for samples in the group specified by label_filter. If zero length then all labels will be included.
`label_factor`	The sample_meta column to use for labelling the samples. If 'rownames' then the rownames will be used.
`label_size`	The text size of the labels.NB ggplot units, not font size units. Default 3.88.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
M = mean_centre() + PCA()
M = model_apply(M,D)
C = pca_scores_plot(factor_name = 'Species')
chart_plot(C,M[2])
```

`pca_scree_plot` *pca_scree_plot_plot class*

Description

Line plot showing percent variance and cumulative percent variance for the computed components.

Usage

```
pca_scree_plot(...)
```

Arguments

`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
C = pca_scree_plot()
```

`permutation_test` *Permutation test class*

Description

Applies a permutation test to a model or model_seq(). The input metric is calculated for all permutations, and can be compared to the results from the unpermuted model to assess model validity.

Usage

```
permutation_test(number_of_permutations = 50, factor_name, ...)
```

Arguments

`number_of_permutations`	The number of permutations to run
`factor_name`	The same of the sample_meta column to use
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
I=permutation_test(factor_name='Species')
```

`permutation_test_plot` *permutation_test_plot class*

Description

Plots the results of a permutation test.

Usage

```
permutation_test_plot(style = "boxplot", binwidth = 0.05, ...)
```

Arguments

`style`	The plot style. One of 'boxplot', 'violin', 'histogram', 'density' or 'scatter'.
`binwidth`	Binwidth for the "histogram" style. Ignored for all other styles.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
C = permutation_test_plot(style='boxplot')
```

`permute_sample_order` *permute_sample_order class*

Description

Permutes the sample order a defined number of times, running the model each time

Usage

```
permute_sample_order(number_of_permutations = 10, ...)
```

Arguments

`number_of_permutations`	The number of times to permute the sample order
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
C = permute_sample_order()
```

`PLSDA` *PLSDA model class*

Description

Partial least squares (PLS) discriminant analysis (DA) model class. This object can be used to train/apply PLS models.

Usage

```
PLSDA(number_components = 2, factor_name, ...)
```

Arguments

`number_components`	The number of PLS components to calculate.
`factor_name`	The sample-meta column name to use.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
M = PLSDA('number_components'=2,factor_name='Species')
```

`plsda_predicted_plot` *plsda_predicted_plot class*

Description

A box plot of the predicted values from a PLSDA model for each class. Only usitable for two class models.

Usage

```
plsda_predicted_plot(factor_name, style = "boxplot", ...)
```

Arguments

`factor_name`	The sample_meta column name to use
`style`	The plot style. One of 'boxplot', 'violin' or 'density'.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
M = mean_centre()+PLSDA(factor_name='Species')
M = model_apply(M,D)

C = plsda_predicted_plot(factor_name='Species')
chart_plot(C,M[2])
```

`plsda_regcoeff_plot` *plsda_regcoeff_plot class*

Description

Plots the regression coefficients of a PLSDA model.

Usage

```
plsda_regcoeff_plot(level, ...)
```

Arguments

`level`	the group label to plot regression coefficients for
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
M = mean_centre()+PLSDA(factor_name='Species')
M = model_apply(M,D)

C = plsda_regcoeff_plot(level='setosa')
chart_plot(C,M[2])
```

`plsda_roc_plot` *plsda_roc_plot class*

Description

Plots the ROC curve of a PLSDA model. Only suitable for two classes.

Usage

```
plsda_roc_plot(factor_name, ...)
```

Arguments

`factor_name`	The sample_meta column name to use
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
M = mean_centre()+PLSDA(factor_name='Species')
M = model_apply(M,D)

C = plsda_roc_plot(factor_name='Species')
chart_plot(C,M[2])
```

`plsda_scores_plot` *plsda_scores_plot class*

Description

2d scatter plot of plsda component scores.

Usage

```
plsda_scores_plot(
  components = c(1, 2),
  points_to_label = "none",
  factor_name,
  ...
)
```

Arguments

`components`	The PLS components to plot (`numeric(2)`)
`points_to_label`	"none", "all", or "outliers" will be labelled on the plot.
`factor_name`	The sample_meta column name for labelling the legend
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
M = mean_centre()+PLSDA(factor_name='Species')
M = model_apply(M,D)

C = plsda_scores_plot(factor_name='Species')
chart_plot(C,M[2])
```

`plsda_vip_plot` *plsda_vip_plot class*

Description

Plots the vip scores of a PLSDA model.

Usage

```
plsda_vip_plot(threshold = 1, level, ...)
```

Arguments

`threshold`	the VIP threshold to plot. Default = 1
`level`	= the group label to plot VIP scores for
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
M = mean_centre()+PLSDA(factor_name='Species')
M = model_apply(M,D)

C = plsda_vip_plot(level='setosa')
chart_plot(C,M[2])
```

PLSR *PLSR model class*

Description

Calculates a PLS regression model using the input data.

Usage

```
PLSR(number_components = 2, factor_name, ...)
```

Arguments

`number_components`	The number of PLS components to calculate.
`factor_name`	The sample_meta column name to use.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
M = PLSR(factor_name='run_order')
```

plsr_cook_dist *plsr_cook_dist class*

Description

Cook's distance for a PLSR model. Cook's distance measures the effect of deleting each observation. Samples with a larger Cook's distance might be considered outlying as they have strong influence on the regression.

Usage

```
plsr_cook_dist(...)
```

Arguments

`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
C = plsr_cook_dist()
```

plsr_prediction_plot *plsr_prediction_plot class*

Description

Plots the true values against the predicted values for a PLSR model.

Usage

```
plsr_prediction_plot(...)
```

Arguments

...	additional slots and values passed to struct_class

Value

struct object

Examples

```
C = plsr_prediction_plot()
```

plsr_qq_plot *plsr_qq_plot class*

Description

Quantiles of PLSR residuals against the qualtiles of a normal distribution

Usage

```
plsr_qq_plot(...)
```

Arguments

...	additional slots and values passed to struct_class

Value

struct object

Examples

```
C = plsr_qq_plot()
```

plsr_residual_hist *plsr_residual_hist class*

Description

A histogram of the model residuals

Usage

```
plsr_residual_hist(...)
```

Arguments

`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
C = plsr_residual_hist()
```

pqn_norm *Probabilistic Quotient Normalisation*

Description

Applies PQN using QC samples as reference samples

Usage

```
pqn_norm(qc_label = "QC", factor_name, ...)
```

Arguments

`qc_label`	= The label for qc samples in the chosen sample_meta column.
`factor_name`	The sample_meta column name containing QC labels.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
M = pqn_norm(factor_name='Species',qc_label='all')
M = model_apply(M,D)
```

`pqn_norm_hist` *plot for PQN normalisation*

Description

plots a histogram of the PQN coeffients

Usage

```
pqn_norm_hist(...)
```

Arguments

`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
C = pqn_norm_hist()
```

`prop_na` *prop_na model class*

Description

Compares proportion of NA for all features in a DatasetExperiment using a Fisher's Exact test

Usage

```
prop_na(alpha = 0.05, mtc = "fdr", factor_name, ...)
```

Arguments

`alpha`	The p-value threshold. Default alpha = 0.05.
`mtc`	Multiple test correction method passed to `p.adjust`. Default mtc = 'fdr'.
`factor_name`	The sample_meta column name to use.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
M = prop_na(factor_name='Species')
```

rsd_filter *rsd filter*

Description

Filters features based on the relative standard deviation (RSD) for the QC samples.

Usage

```
rsd_filter(rsd_threshold = 20, qc_label = "QC", factor_name, ...)
```

Arguments

rsd_threshold	Features with RSD greater than the threshold are removed.
qc_label	The label used to identify QC samples in the chosen sample_meta column.
factor_name	The name of the sample_meta column containing QC labels.
...	additional slots and values passed to struct_class

Value

struct object

Examples

```
M = rsd_filter(factor_name='class')
```

rsd_filter_hist *plot for rsd filter*

Description

plots a histogram of the calculated RSD for the RSD filter

Usage

```
rsd_filter_hist(...)
```

Arguments

...	additional slots and values passed to struct_class

Value

struct object

Examples

```
C = rsd_filter_hist()
```

run,bootstrap,DatasetExperiment,metric-method

Runs an iterator, applying the chosen model multiple times.

Description

Running an iterator will apply the iterator a number of times to a DatasetExperiment. For example, in cross-validation the same model is applied multiple times to the same data, splitting it into training and test sets. The input metric object can be calculated and collected for each iteration as an output.

Usage

```
## S4 method for signature 'bootstrap,DatasetExperiment,metric'
run(I, D, MET = NULL)

## S4 method for signature 'forward_selection_byrank,DatasetExperiment,metric'
run(I, D, MET)

## S4 method for signature 'grid_search_1d,DatasetExperiment,metric'
run(I, D, MET)

## S4 method for signature 'kfold_xval,DatasetExperiment,metric'
run(I, D, MET = NULL)

## S4 method for signature 'permutation_test,DatasetExperiment,metric'
run(I, D, MET = NULL)

## S4 method for signature 'permute_sample_order,DatasetExperiment,metric'
run(I, D, MET)
```

Arguments

`I`	an iterator object
`D`	a DatasetExperiment object
`MET`	a metric object

Value

Modified iterator object

Examples

```
D = iris_DatasetExperiment() # get some data
MET = metric()  # use a metric
I = example_iterator() # initialise iterator
models(I) = example_model() # set the model
I = run(I,D,MET) # run
```

r_squared *Coefficient of determination class*

Description

Coefficient of determination (r-squared).

Usage

```
r_squared(...)
```

Arguments

`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
MET = r_squared()
```

sb_corr *sbcms*

Description

Signal/batch correction using SMCBMS package

Usage

```
sb_corr(
  order_col,
  batch_col,
  qc_col,
  smooth = 0,
  use_log = TRUE,
  min_qc = 4,
  qc_label = "QC",
  ...
)
```

Arguments

`order_col`	The sample-meta column containing the order of measurement.
`batch_col`	The sample_meta column containing the batch labels.
`qc_col`	The sample_meta column containing QC labels.
`smooth`	Spline smoothing parameter. Should be in the range 0 to 1. If set to 0 it will be estimated using leave-one-out cross-validation.
`use_log`	Perform the signal correction fit on the log scaled data. Default is TRUE.
`min_qc`	Minimum number of measured quality control (QC) samples required for signal correction within feature per batch. Default 4.
`qc_label`	The label used in `qc_col` to identify QC samples.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
M = sb_corr(order_col='run_order',batch_col='batch_no',qc_col='class')
```

`split_data` *Split data into subsets*

Description

Splits the data into a training and test set.

Usage

```
split_data(p_train, ...)
```

Arguments

`p_train`	The proportion of samples in the training set.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
M = split_data(p_train=0.75)
```

`stratified_split` *Stratified sampling*

Description

Splits the data into a training and test set, using stratification to keep group sizes in equal proportions to the full dataset.

Usage

```
stratified_split(p_train, factor_name, ...)
```

Arguments

`p_train`	The proportion of samples in the training set.
`factor_name`	The column of sample_meta to use for stratification
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
M = stratified_split(p_train=0.75,factor_name='Species')
M = model_apply(M,D)
```

`structToolbox` *structToolbox: Examples of tools built using the Statistics in R Using Class Templates (struct) package*

Description

This package extends the classes defined in the struct package

SVM *SVM model classifier*

Description

Support vector machines model classifier. Wraps svm from the "e1071" package, which interfaces with the "libsvm" library to train SVM classifiers.

Usage

```
SVM(
  factor_name,
  kernel = "linear",
  degree = 3,
  gamma = 1,
  coef0 = 0,
  cost = 1,
  class_weights = NULL,
  ...
)
```

Arguments

`factor_name`	The sample-meta column name to use for group labels
`kernel`	the kernel used in training and predicting. You might consider changing some of the following parameters, depending on the kernel type. **linear:** $u'v$ **polynomial:** $(\gamma u'v + coef0)^{degree}$ **radial basis:** $e^(- \gamma|u - v|^2)$ **sigmoid:** $tanh(\gamma u'v + coef0)$
`degree`	parameter needed for kernel of type `polynomial` (default: 3)
`gamma`	parameter needed for all kernels except `linear` (default: 1/(data dimension))
`coef0`	parameter needed for kernels of type `polynomial` and `sigmoid` (default: 0)
`cost`	cost of constraints violation (default: 1)—it is the ‘C’-constant of the regularization term in the Lagrange formulation.
`class_weights`	a named vector of weights for the different classes, used for asymmetric class sizes. Not all factor levels have to be supplied (default weight: 1). All components have to be named. Specifying "inverse" will choose the weights inversely proportional to the class distribution.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
M = SVM(factor_name='Species',gamma=1)
```

svm_plot_2d *SVM boundary plot (2d)*

Description

Plots the training data and the SVM boundary. 2d data only (ncol(D$data)==2).

Usage

```
svm_plot_2d(factor_name, npoints = 100, ...)
```

Arguments

`factor_name`	The column of sample_meta to use
`npoints`	Used to control the resolution of the grid used to plot the boundary. Default 100.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
D = iris_DatasetExperiment()
M = filter_smeta(mode='exclude',levels='setosa',factor_name='Species') +
    mean_centre()+PCA(number_components=2)+
    SVM(factor_name='Species',kernel='linear')
M = model_apply(M,D)

C = svm_plot_2d(factor_name='Species')
chart_plot(C,M[4],predicted(M[3]))
```

tSNE *tSNE method class*

Description

t-Distributed Stochastic Neighbor Embedding (tSNE) class. This object can be used to train/apply tSNE models to DatasetExperiment objects.

Usage

```
tSNE(
  dims = 2,
  perplexity = 30,
  max_iter = 100,
  theta = 0.5,
  check_duplicates = FALSE,
  init = NULL,
  eta = 200,
  ...
)
```

Arguments

`dims`	integer; Output dimensionality (default: 2)
`perplexity`	numeric; Perplexity parameter (should not be bigger than 3 * perplexity < nrow(X) - 1, see details for interpretation)
`max_iter`	integer; Number of iterations (default: 1000)
`theta`	numeric; Speed/accuracy trade-off (increase for less accuracy), set to 0.0 for exact TSNE (default: 0.5)
`check_duplicates`	logical; Checks whether duplicates are present. It is best to make sure there are no duplicates present and set this option to FALSE, especially for large datasets (default: TRUE)
`init`	Initial locations of the objects. If NULL, random initialization will be used. If NULL then
`eta`	numeric; Learning rate (default: 200.0)
`...`	additional slots and values passed to struct_class

Details

This object is a wrapper for Rtsne::Rtsne.

Value

struct object

Examples

```
M = tSNE()
```

`tSNE_scatter` *tSNE_scatter class*

Description

plots the new representation of data after applying tSNE

Usage

```
tSNE_scatter(factor_name, ...)
```

Arguments

`factor_name`	Sample_meta column named used for colouring the points.
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
M = tSNE_scatter(factor_name='Species')
```

---

ttest *t-test model class*

Description

t-test model class. Calculate t-test for all features in a DatasetExperiment.

Usage

```
ttest(
  alpha = 0.05,
  mtc = "fdr",
  factor_names,
  paired = FALSE,
  paired_factor = character(0),
  ...
)
```

Arguments

`alpha`	The p-value threshold. Default alpha = 0.05.
`mtc`	Multiple test correction method passed to `p.adjust`. Default mtc = 'fdr'.
`factor_names`	The sample_meta column name to use.
`paired`	TRUE or FALSE to use a paired t-test.
`paired_factor`	The name of the sample_meta column used to indicate which samples are from the same subject. Must be provided if `paired` = `TRUE`
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
M = ttest(factor_name='class')
```

`vec_norm` *Vector normalisation*

Description

Applies vector normalisation, such the sum of squared values for each sample after normalisation are equal to 1.

Usage

```
vec_norm(...)
```

Arguments

`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
M = vec_norm()
```

`wilcox_p_hist` *plot histogram of p values*

Description

plots a histogram of p values

Usage

```
wilcox_p_hist(...)
```

Arguments

`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
M = wilcox_p_hist()
```

wilcox_test *Wilcoxon signed rank test method class*

Description

Calculates a signed rank test for all features in a DatasetExperiment. Used as a non-parametric ttest.

Usage

```
wilcox_test(
  alpha = 0.05,
  mtc = "fdr",
  factor_names,
  paired = FALSE,
  paired_factor = character(0),
  ...
)
```

Arguments

`alpha`	The p-value threshold. Default alpha = 0.05.
`mtc`	Multiple test correction method passed to `p.adjust`. Default mtc = 'fdr'.
`factor_names`	The sample_meta column name to use.
`paired`	TRUE or FALSE to use a paired test.
`paired_factor`	The name of the sample_meta column used to indicate which samples are from the same subject. Must be provided if `paired = TRUE`
`...`	additional slots and values passed to struct_class

Value

struct object

Examples

```
M = wilcox_test(factor_name='class')
```

Index