

Package ‘promor’

July 23, 2025

Type Package

Title Proteomics Data Analysis and Modeling Tools

Version 0.2.1

Description A comprehensive, user-friendly package for label-free proteomics data analysis and machine learning-based modeling. Data generated from 'MaxQuant' can be easily used to conduct differential expression analysis, build predictive models with top protein candidates, and assess model performance. promor includes a suite of tools for quality control, visualization, missing data imputation (Lazar et. al. (2016) <[doi:10.1021/acs.jproteome.5b00981](https://doi.org/10.1021/acs.jproteome.5b00981)>), differential expression analysis (Ritchie et. al. (2015) <[doi:10.1093/nar/gkv007](https://doi.org/10.1093/nar/gkv007)>), and machine learning-based modeling (Kuhn (2008) <[doi:10.18637/jss.v028.i05](https://doi.org/10.18637/jss.v028.i05)>).

License LGPL (>= 2.1)

Encoding UTF-8

Language en-US

RoxygenNote 7.2.2

VignetteBuilder knitr

Suggests covr, knitr, rmarkdown, testthat (>= 3.0.0)

Depends R (>= 3.5.0)

URL <https://github.com/caranathunge/promor>,
<https://caranathunge.github.io/promor/>

Imports reshape2, ggplot2, ggrepel, gridExtra, limma, statmod,
pcaMethods, VIM, missForest, caret, kernlab, xgboost,
naivebayes, viridis, pROC

LazyData true

Config/testthat/edition 3

BugReports <https://github.com/caranathunge/promor/issues>

NeedsCompilation no

Author Chathurani Ranathunge [aut, cre, cph] (ORCID:
<<https://orcid.org/0000-0003-1901-2119>>)

Maintainer Chathurani Ranathunge <caranathunge86@gmail.com>

Repository CRAN

Date/Publication 2023-07-17 22:30:02 UTC

Contents

aver_techreps	2
corr_plot	3
covid_fit_df	5
covid_norm_df	5
create_df	6
ecoli_fit_df	8
ecoli_norm_df	9
feature_plot	9
filterbygroup_na	11
find_dep	12
heatmap_de	14
heatmap_na	16
impute_na	18
impute_plot	20
normalize_data	22
norm_plot	24
onegroup_only	25
performance_plot	27
pre_process	29
rem_feature	30
rem_sample	32
roc_plot	33
split_data	35
test_models	36
train_models	38
varimp_plot	40
volcano_plot	42
Index	44

aver_techreps	<i>Compute average intensity</i>
---------------	----------------------------------

Description

This function computes average intensities across technical replicates for each sample.

Usage

```
aver_techreps(raw_df)
```

Arguments

raw_df A raw_df object containing technical replicates.

Details

aver_techreps assumes that column names in the data frame follow the "Group_UniqueSampleID_TechnicalReplicate" notation. (Use head(raw_df) to see the structure of the raw_df object.)

Value

A raw_df object of averaged intensities.

Author(s)

Chathurani Ranathunge

See Also

- [avearrays](#) from [limma](#) package.
- [create_df](#)

Examples

```
## Use a data set containing technical replicates to create a raw_df object
raw_df <- create_df(
  prot_groups = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/pg2.txt",
  exp_design = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/ed2.txt",
  tech_reps = TRUE
)

# Compute average intensities across technical replicates.
rawdf_ave <- aver_techreps(raw_df)
```

corr_plot

Correlation between technical replicates

Description

This function generates scatter plots to visualize the correlation between a given pair of technical replicates (Eg: 1 vs 2) for each sample.

Usage

```
corr_plot(
  raw_df,
  rep_1,
  rep_2,
  save = FALSE,
  file_type = "pdf",
  palette = "viridis",
  text_size = 5,
```

```
n_row = 4,  
n_col = 4,  
dpi = 80,  
file_path = NULL  
)
```

Arguments

raw_df	A raw_df object (output of create_df) containing technical replicates.
rep_1	Numerical. Technical replicate number.
rep_2	Numerical. Number of the second technical replicate to compare to rep1.
save	Logical. If TRUE saves a copy of the plot in the directory provided in file_path.
file_type	File type to save the scatter plots. Default is "pdf".
palette	Viridis color palette option for plots. Default is "viridis". See viridis for available options.
text_size	Text size for plot labels, axis labels etc. Default is 10.
n_row	Numerical. Number of plots to print in a row in a single page. Default is 4.
n_col	Numerical. Number of plots to print in a column in a single page. Default is 4.
dpi	Plot resolution. Default is 80.
file_path	A string containing the directory path to save the file.

Details

- Given a data frame of log-transformed intensities (a raw_df object) and a pair of numbers referring to the technical replicates, corr_plot produces a list of scatter plots showing correlation between the given pair of technical replicates for all the samples provided in the data frame.
- Note: $n_row * n_col$ should be equal to the number of samples to display in a single page.

Value

A list of ggplot2 plot objects.

Author(s)

Chathurani Ranathunge

See Also

[create_df](#)

Examples

```
## Use a data set containing technical replicates to create a raw_df object
raw_df <- create_df(
  prot_groups = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/pg2.txt",
  exp_design = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/ed2.txt",
  tech_reps = TRUE
)
```

```
## Compare technical replicates 1 vs. 2 for all samples
corr_plot(raw_df, rep_1 = 1, rep_2 = 2)
```

covid_fit_df	<i>Suvarna et al 2021 LFQ data (fit object)</i>
--------------	---

Description

An object of class "MArrayLM" from running find_dep on covid_norm_df

Usage

```
data(covid_fit_df)
```

Format

An object of class "MArrayLM"

References

<https://www.frontiersin.org/articles/10.3389/fphys.2021.652799/full#h3>

covid_norm_df	<i>Suvarna et al 2021 LFQ data (normalized)</i>
---------------	---

Description

A dataframe containing normalized LFQ protein intensity data for 230 proteins in 35 samples (a subset of the original data set)

Usage

```
data(covid_norm_df)
```

Format

A data frame with 230 rows (proteins) and 35 columns (samples)

References

<https://www.frontiersin.org/articles/10.3389/fphys.2021.652799/full#h3>

create_df	<i>Create a data frame of protein intensities</i>
-----------	---

Description

This function creates a data frame of protein intensities

Usage

```
create_df(
  prot_groups,
  exp_design,
  input_type = "MaxQuant",
  data_type = "LFQ",
  filter_na = TRUE,
  filter_prot = TRUE,
  uniq_pep = 2,
  tech_reps = FALSE,
  zero_na = TRUE,
  log_tr = TRUE,
  base = 2
)
```

Arguments

prot_groups	File path to a proteinGroups.txt file produced by MaxQuant or a standard input file containing a quantitative matrix where the proteins or protein groups are indicated by rows and the samples by columns.
exp_design	File path to a text file containing the experimental design.
input_type	Type of input file indicated by prot_groups. Available options are: "MaxQuant", if a proteinGroups.txt file is used, or "standard" if a standard input file is used. Default is "MaxQuant."
data_type	Type of sample protein intensity data columns to use from the proteinGroups.txt file. Some available options are "LFQ", "iBAQ", "Intensity". Default is "LFQ." User-defined prefixes in the proteinGroups.txt file are also allowed. The data_type argument is case-sensitive, and only applies when input_type = "MaxQuant".
filter_na	Logical. If TRUE(default), filters out empty rows and columns from the data frame.

filter_prot	Logical. If TRUE (default), filters out reverse proteins, proteins only identified by site, potential contaminants, and proteins identified with less than the minimum number of unique peptides indicated by uniq_pep. Only applies when input_type = "MaxQuant".
uniq_pep	Numerical. Proteins that are identified by this number or fewer number of unique peptides are filtered out (default is 2). Only applies when input_type = "MaxQuant".
tech_reps	Logical. Indicate as TRUE if technical replicates are present in the data. Default is FALSE.
zero_na	Logical. If TRUE (default), zeros are considered missing values and replaced with NAs.
log_tr	Logical. If TRUE (default), intensity values are log transformed to the base indicated by base.
base	Numerical. Logarithm base. Default is 2.

Details

- This function first reads in the proteinGroups.txt file produced by MaxQuant or a standard input file containing a quantitative matrix where the proteins or protein groups are indicated by rows and the samples by columns.
- It then reads in the expDesign.txt file provided as exp_design and extracts relevant information from it to add to the data frame. an example of the expDesign.txt is provided here: https://raw.githubusercontent.com/caranathunge/promor_example_data/main/ed1.txt.
- First, empty rows and columns are removed from the data frame.
- Next, if a proteinGroups.txt file is used, it filters out reverse proteins, proteins that were only identified by site, and potential contaminants. Then it removes proteins identified with less than the number of unique peptides indicated by uniq_pep from the data frame.
- Next, it extracts the intensity columns indicated by data type and the selected protein rows from the data frame.
- Converts missing values (zeros) to NAs.
- Finally, the function log transforms the intensity values.

Value

A raw_df object which is a data frame containing protein intensities. Proteins or protein groups are indicated by rows and samples by columns.

Author(s)

Chathurani Ranathunge

Examples

```
### Using a proteinGroups.txt file produced by MaxQuant as input.
## Generate a raw_df object with default settings. No technical replicates.
```

```

raw_df <- create_df(
  prot_groups = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/pg1.txt",
  exp_design = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/ed1.txt",
  input_type = "MaxQuant"
)

## Data containing technical replicates
raw_df <- create_df(
  prot_groups = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/pg2.txt",
  exp_design = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/ed2.txt",
  input_type = "MaxQuant",
  tech_reps = TRUE
)

## Alter the number of unique peptides needed to retain a protein
raw_df <- create_df(
  prot_groups = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/pg1.txt",
  exp_design = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/ed1.txt",
  input_type = "MaxQuant",
  uniq_pep = 1
)

## Use "iBAQ" values instead of "LFQ" values
raw_df <- create_df(
  prot_groups = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/pg1.txt",
  exp_design = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/ed1.txt",
  input_type = "MaxQuant",
  data_type = "iBAQ"
)

### Using a universal standard input file instead of MaxQuant output.
raw_df <- create_df(
  prot_groups = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/st.txt",
  exp_design = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/ed1.txt",
  input_type = "standard"
)

```

ecoli_fit_df

Cox et al 2014 LFQ data (fit object)

Description

An object of class "MArrayLM" from running find_dep on ecoli_norm_df

Usage

```
data(ecoli_fit_df)
```

Format

An object of class "MArrayLM"

References

<https://europepmc.org/article/MED/24942700#id609082>

ecoli_norm_df	<i>Cox et al 2014 LFQ data (normalized)</i>
---------------	---

Description

A dataframe containing normalized LFQ protein intensity data for 4360 proteins in 6 samples

Usage

```
data(ecoli_norm_df)
```

Format

A data frame with 4360 rows (proteins) and 6 columns (samples)

References

<https://europepmc.org/article/MED/24942700#id609082>

feature_plot	<i>Visualize feature (protein) variation among conditions</i>
--------------	---

Description

This function visualizes protein intensity differences among conditions (classes) using box plots or density distribution plots.

Usage

```
feature_plot(  
  model_df,  
  type = "box",  
  text_size = 10,  
  palette = "viridis",  
  n_row,  
  n_col,  
  save = FALSE,  
  file_path = NULL,  
  file_name = "Feature_plot",
```

```
file_type = "pdf",  
dpi = 80,  
plot_width = 7,  
plot_height = 7  
)
```

Arguments

model_df	A model_df object from performing pre_process.
type	Type of plot to generate. Choices are "box" or "density." Default is "box."
text_size	Text size for plot labels, axis labels etc. Default is 10.
palette	Viridis color palette option for plots. Default is "viridis". See viridis for available options.
n_row	Number of rows to print the plots.
n_col	Number of columns to print the plots.
save	Logical. If TRUE saves a copy of the plot in the directory provided in file_path.
file_path	A string containing the directory path to save the file.
file_name	File name to save the plot. Default is "Feature_plot."
file_type	File type to save the plot. Default is "pdf".
dpi	Plot resolution. Default is 80.
plot_width	Width of the plot. Default is 7.
plot_height	Height of the plot. Default is 7.

Details

This function visualizes condition-wise differences in protein intensity using boxplots and/or density plots.

Value

A ggplot2 object

Author(s)

Chathurani Ranathunge

See Also

- pre_process, rem_feature

Examples

```
## Create a model_df object with default settings.
covid_model_df <- pre_process(covid_fit_df, covid_norm_df)

## Feature variation - box plots
feature_plot(covid_model_df, type = "box", n_row = 4, n_col = 2)

## Density plots
feature_plot(covid_model_df, type = "density")

## Change color palette
feature_plot(covid_model_df, type = "density", n_row = 4, n_col = 2, palette = "rocket")
```

filterbygroup_na	<i>Filter proteins by group level missing data</i>
------------------	--

Description

This function filters out proteins based on missing data at the group level.

Usage

```
filterbygroup_na(raw_df, set_na = 0.34, filter_condition = "either")
```

Arguments

raw_df	A raw_df object (output of create_df)
set_na	The proportion of missing data allowed. Default is 0.34 (one third of the samples in the group).
filter_condition	If set to "each", proteins that exceed the missing value proportion threshold set by set_na in each group will be removed (lenient). If set to "either" (default), proteins that exceed the missing value proportion threshold set by set_na in at least one group will be removed (stringent).

Details

- This function first extracts group or condition information from the raw_df object and assigns samples to their groups.
- If filter_condition = "each", it then removes proteins (rows) from the data frame if the proportion of NAs in **each** group exceeds the threshold indicated by set_na (default is 0.34). This option is more lenient in comparison to filter_condition = "either", where proteins that exceeds the missing data threshold in **either** group gets removed from the data frame.

Value

A raw_df object.

Author(s)

Chathurani Ranathunge

See Also

[create_df](#)

Examples

```
# Generate a raw_df object with default settings. No technical replicates.
raw_df <- create_df(
  prot_groups = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/pg1.txt",
  exp_design = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/ed1.txt"
)

## Remove proteins that exceed 34% NAs in either group (default)
rawdf_filt1 <- filterbygroup_na(raw_df)

## Remove proteins that exceed 34% NAs in each group
rawdf_filt2 <- filterbygroup_na(raw_df, filter_condition = "each")

## Proportion of samples with NAs allowed in each group = 0.5
rawdf_filt3 <- filterbygroup_na(raw_df, set_na = 0.5, filter_condition = "each")
```

find_dep

Identify differentially expressed proteins between groups

Description

This function performs differential expression analysis on protein intensity data with limma.

Usage

```
find_dep(
  df,
  save_output = FALSE,
  save_tophits = FALSE,
  file_path = NULL,
  adj_method = "BH",
  cutoff = 0.05,
  lfc = 1,
  n_top = 20
)
```

Arguments

<code>df</code>	A <code>norm_df</code> object or an <code>imp_df</code> object.
<code>save_output</code>	Logical. If TRUE saves results from the differential expression analysis in a text file labeled "limma_output.txt" in the directory specified by <code>file_path</code> .
<code>save_tophits</code>	Logical. If TRUE saves <code>n_top</code> number of top hits from the differential expression analysis in a text file labeled "TopHits.txt" in the directory specified by <code>file_path</code> .
<code>file_path</code>	A string containing the directory path to save the file.
<code>adj_method</code>	Method used for adjusting the p-values for multiple testing. Default is "BH" for "Benjamini-Hochberg" method.
<code>cutoff</code>	Cutoff value for p-values and adjusted p-values. Default is 0.05.
<code>lfc</code>	Minimum absolute log2-fold change to use as threshold for differential expression.
<code>n_top</code>	The number of top differentially expressed proteins to save in the "TopHits.txt" file. Default is 20.

Details

- It is important that the data is first log-transformed, ideally, imputed, and normalized before performing differential expression analysis.
- `save_output` saves the complete results table from the differential expression analysis.
- `save_tophits` first subsets the results to those with absolute log fold change of more than 1, performs multiple correction with the method specified in `adj_method` and outputs the top `n_top` results based on lowest p-value and adjusted p-value.
- If the number of hits with absolute log fold change of more than 1 is less than `n_top`, `find_dep` prints only those with log-fold change > 1 to "TopHits.txt".
- If the `file_path` is not specified, text files will be saved in a temporary directory.

Value

A `fit_df` object, which is similar to a `limma fit` object.

Author(s)

Chathurani Ranathunge

References

Ritchie, Matthew E., et al. "limma powers differential expression analyses for RNA-sequencing and microarray studies." *Nucleic acids research* 43.7 (2015): e47-e47.

See Also

- `normalize_data`
- `lmFit`, `eBayes`, `topTable`, and `write.fit` functions from the `limma` package.

Examples

```
## Perform differential expression analysis using default settings
fit_df1 <- find_dep(ecoli_norm_df)

## Change p-value and adjusted p-value cutoff
fit_df2 <- find_dep(ecoli_norm_df, cutoff = 0.1)
```

heatmap_de

Heatmap of differentially expressed proteins

Description

This function generates a heatmap to visualize differentially expressed proteins between groups

Usage

```
heatmap_de(
  fit_df,
  df,
  adj_method = "BH",
  cutoff = 0.05,
  lfc = 1,
  sig = "adjP",
  n_top = 20,
  palette = "viridis",
  text_size = 10,
  save = FALSE,
  file_path = NULL,
  file_name = "HeatmapDE",
  file_type = "pdf",
  dpi = 80,
  plot_height = 7,
  plot_width = 7
)
```

Arguments

fit_df	A fit_df object from performing find_dep.
df	The norm_df object or the imp_df object from which the fit_df object was obtained.
adj_method	Method used for adjusting the p-values for multiple testing. Default is "BH".
cutoff	Cutoff value for p-values and adjusted p-values. Default is 0.05.
lfc	Minimum absolute log2-fold change to use as threshold for differential expression. Default is 1.

sig	Criteria to denote significance. Choices are "adjP" (default) for adjusted p-value or "P" for p-value.
n_top	Number of top hits to include in the heat map.
palette	Viridis color palette option for plots. Default is "viridis". See viridis for available options.
text_size	Text size for axis text, labels etc.
save	Logical. If TRUE saves a copy of the plot in the directory provided in file_path.
file_path	A string containing the directory path to save the file.
file_name	File name to save the plot. Default is "HeatmapDE."
file_type	File type to save the plot. Default is "pdf".
dpi	Plot resolution. Default is 80.
plot_height	Height of the plot. Default is 7.
plot_width	Width of the plot. Default is 7.

Details

By default the tiles in the heatmap are reordered by intensity values along both axes (x axis = samples, y axis = proteins).

Value

A ggplot2 plot object.

Author(s)

Chathurani Ranathunge

See Also

- `find_dep`
- `topTable` and `lmFit` functions from the [limma](#) package.

Examples

```
## Build a heatmap of differentially expressed proteins using the provided
## example fit_df and norm_df data objects
heatmap_de(covid_fit_df, covid_norm_df)

## Create a heatmap with P-value of 0.05 and log fold change of 1 as
## significance criteria.
heatmap_de(covid_fit_df, covid_norm_df, cutoff = 0.05, sig = "P")

## Visualize the top 30 differentially expressed proteins in the heatmap and
## change the color palette
heatmap_de(covid_fit_df, covid_norm_df,
  cutoff = 0.05, sig = "P", n_top = 30,
  palette = "magma"
)
```

heatmap_na

*Visualize missing data***Description**

This function visualizes the patterns of missing value occurrence using a heatmap.

Usage

```
heatmap_na(
  raw_df,
  protein_range,
  sample_range,
  reorder_x = FALSE,
  reorder_y = FALSE,
  x_fun = mean,
  y_fun = mean,
  palette = "viridis",
  label_proteins = FALSE,
  text_size = 10,
  save = FALSE,
  file_type = "pdf",
  file_path = NULL,
  file_name = "Missing_data_heatmap",
  plot_width = 15,
  plot_height = 15,
  dpi = 80
)
```

Arguments

raw_df	A raw_df object (output from create_df).
protein_range	The range or subset of proteins (rows) to plot. If not provided, all the proteins (rows) in the data frame will be used.
sample_range	The range of samples to plot. If not provided, all the samples (columns) in the data frame will be used.
reorder_x	Logical. If TRUE samples on the x axis are reordered using the function given in x_fun. Default is FALSE.
reorder_y	Logical. If TRUE proteins in the y axis are reordered using the function given in y_fun. Default is FALSE.
x_fun	Function to reorder samples along the x axis. Possible options are mean and sum. Default is mean.
y_fun	Function to reorder proteins along the y axis. Possible options are mean and sum. Default is mean.

palette	Viridis color palette option for plots. Default is "viridis". See viridis for available options.
label_proteins	If TRUE proteins on the y axis will be labeled with their Majority Protein IDs. Default is FALSE.
text_size	Text size for axis labels. Default is 10.
save	Logical. If TRUE saves a copy of the plot in the directory provided in file_path.
file_type	File type to save the heatmap. Default is "pdf".
file_path	A string containing the directory path to save the file.
file_name	File name to save the heatmap. Default is "Missing_data_heatmap".
plot_width	Width of the plot. Default is 15.
plot_height	Height of the plot. Default is 15.
dpi	Plot resolution. Default is 80.

Details

This function visualizes patterns of missing value occurrence using a heatmap. The user can choose to reorder the axes using the available functions (x_fun, y_fun) to better understand the underlying cause of missing data.

Value

A ggplot2 plot object.

Author(s)

Chathurani Ranathunge

See Also

[create_df](#)

Examples

```
## Generate a raw_df object with default settings. No technical replicates.
raw_df <- create_df(
  prot_groups = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/pg1.txt",
  exp_design = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/ed1.txt"
)

## Missing data heatmap with default settings.
heatmap_na(raw_df)

## Missing data heatmap with x and y axes reordered by the mean (default) of
## protein intensity.
heatmap_na(raw_df,
  reorder_x = TRUE, reorder_y = TRUE
)
```

```
## Missing data heatmap with x and y axes reordered by the sum of
## protein intensity.
heatmap_na(raw_df,
  reorder_x = TRUE, reorder_y = TRUE, x_fun = sum,
  y_fun = sum
)

## Missing data heatmap for a subset of the proteins with x and y axes
## reordered by the mean (default) of protein intensity and the y axis
## labeled with protein IDs.
heatmap_na(raw_df,
  protein_range = 1:30,
  reorder_x = TRUE, reorder_y = TRUE,
  label_proteins = TRUE
)
```

impute_na

Impute missing values

Description

This function imputes missing values using a user-specified imputation method.

Usage

```
impute_na(
  df,
  method = "minProb",
  tune_sigma = 1,
  q = 0.01,
  maxiter = 10,
  ntree = 20,
  n_pcs = 2,
  seed = NULL
)
```

Arguments

df	A raw_df object (output of create_df) containing missing values or a norm_df object after performing normalization.
method	Imputation method to use. Default is "minProb". Available methods: "minDet", "RF", "kNN", and "SVD".
tune_sigma	A scalar used in the "minProb" method for controlling the standard deviation of the Gaussian distribution from which random values are drawn for imputation. Default is 1.
q	A scalar used in "minProb" and "minDet" methods to obtain a low intensity value for imputation. q should be set to a very low value. Default is 0.01.

maxiter	Maximum number of iterations to be performed when using the "RF" method. Default is 10.
ntree	Number of trees to grow in each forest when using the "RF" method. Default is 20.
n_pcs	Number of principal components to calculate when using the "SVD" method. Default is 2.
seed	Numerical. Random number seed. Default is NULL

Details

- Ideally, you should first remove proteins with high levels of missing data using the `filterbygroup_na` function before running `impute_na` on the `raw_df` object or the `norm_df` object.
- `impute_na` function imputes missing values using a user-specified imputation method from the available options, `minProb`, `minDet`, `kNN`, `RF`, and `SVD`.
- **Note: Some imputation methods may require that the data be normalized prior to imputation.**
- Make sure to fix the random number seed with `seed` for reproducibility

Value

An `imp_df` object, which is a data frame of protein intensities with no missing values.

Author(s)

Chathurani Ranathunge

References

Lazar, Cosmin, et al. "Accounting for the multiple natures of missing values in label-free quantitative proteomics data sets to compare imputation strategies." *Journal of proteome research* 15.4 (2016): 1116-1125.

See Also

More information on the available imputation methods can be found in their respective packages.

- [create_df](#)
- For `minProb` and `minDet` methods, see `imputeLCMD` package.
- For Random Forest (RF) method, see [missForest](#).
- For `kNN` method, see [kNN](#) from the `VIM` package.
- For `SVD` method, see [pca](#) from the `pcaMethods` package.

Examples

```
## Generate a raw_df object with default settings. No technical replicates.
raw_df <- create_df(
  prot_groups = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/pg1.txt",
  exp_design = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/ed1.txt"
)

## Impute missing values in the data frame using the default minProb
## method.
imp_df1 <- impute_na(raw_df, seed = 3312)

## Impute using the RF method with the number of iterations set at 5
## and number of trees set at 100.
imp_df2 <- impute_na(raw_df,
  method = "RF",
  maxiter = 5, ntree = 100,
  seed = 3312
)

## Using the kNN method.
imp_df3 <- impute_na(raw_df, method = "kNN", seed = 3312)

## Using the SVD method with n_pcs set to 3.
imp_df4 <- impute_na(raw_df, method = "SVD", n_pcs = 3, seed = 3312)

## Using the minDet method with q set at 0.001.
imp_df5 <- impute_na(raw_df, method = "minDet", q = 0.001, seed = 3312)

## Impute a normalized data set using the kNN method
imp_df6 <- impute_na(ecoli_norm_df, method = "kNN")
```

impute_plot

Visualize the impact of imputation

Description

This function generates density plots to visualize the impact of missing data imputation on the data.

Usage

```
impute_plot(
  original,
  imputed,
  global = TRUE,
```

```

    text_size = 10,
    palette = "viridis",
    n_row,
    n_col,
    save = FALSE,
    file_path = NULL,
    file_name = "Impute_plot",
    file_type = "pdf",
    plot_width = 7,
    plot_height = 7,
    dpi = 80
)

```

Arguments

original	A raw_df object (output of create_df) containing missing values or a norm_df object containing normalized protein intensity data.
imputed	An imp_df object obtained from running impute_na on the same data frame provided as original.
global	Logical. If TRUE (default), a global density plot is produced. If FALSE, sample-wise density plots are produced.
text_size	Text size for plot labels, axis labels etc. Default is 10.
palette	Viridis color palette option for plots. Default is "viridis". See viridis for available options.
n_row	Used if global = FALSE to indicate the number of rows to print the plots.
n_col	Used if global = FALSE to indicate the number of columns to print the plots.
save	Logical. If TRUE saves a copy of the plot in the directory provided in file_path.
file_path	A string containing the directory path to save the file.
file_name	File name to save the density plot/s. Default is "Impute_plot."
file_type	File type to save the density plot/s. Default is "pdf".
plot_width	Width of the plot. Default is 7.
plot_height	Height of the plot. Default is 7.
dpi	Plot resolution. Default is 80.

Details

- Given two data frames, one with missing values and the other, an imputed data frame (imp_df object) of the same data set, impute_plot generates global or sample-wise density plots to visualize the impact of imputation on the data set.
- Note, when sample-wise option is selected (global = FALSE), n_col and n_row can be used to specify the number of columns and rows to print the plots.
- If you choose to specify n_row and n_col, make sure that $n_row * n_col$ matches the total number of samples in the data frame.

Value

A ggplot2 plot object.

Author(s)

Chathurani Ranathunge

Examples

```
## Generate a raw_df object with default settings. No technical replicates.
raw_df <- create_df(
  prot_groups = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/pg1.txt",
  exp_design = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/ed1.txt"
)

## Impute missing values in the data frame using the default minProb
## method.
imp_df <- impute_na(raw_df)

## Visualize the impact of missing data imputation with a global density
## plot.
impute_plot(original = raw_df, imputed = imp_df)

## Make sample-wise density plots
impute_plot(raw_df, imp_df, global = FALSE)

## Print plots in user-specified numbers of rows and columns
impute_plot(raw_df, imp_df, global = FALSE, n_col = 2, n_row = 3)
```

normalize_data

Normalize intensity data

Description

This function normalizes data using a user-specified normalization method.

Usage

```
normalize_data(df, method = "quantile")
```

Arguments

df	An imp_df object with missing values imputed using impute_na or a raw_df object containing missing values.
method	Name of the normalization method to use. Choices are "none", "scale", "quantile" or "cyclicloess." Default is "quantile."

Details

- `normalize_data` is a wrapper function around the [normalizeBetweenArrays](#) function from the `limma` package.
- This function normalizes intensity values to achieve consistency among samples.
- It assumes that the intensities in the data frame have been log-transformed, therefore, it is important to make sure that `create_df` was run with `log_tr = TRUE`(default) when creating the `raw_df` object.

Value

A `norm_df` object, which is a data frame of normalized protein intensities.

Author(s)

Chathurani Ranathunge

See Also

- `create_df`
- `impute_na`
- See [normalizeBetweenArrays](#) in the R package `limma` for more information on the different normalization methods available.

Examples

```
## Generate a raw_df object with default settings. No technical replicates.
raw_df <- create_df(
  prot_groups = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/pg1.txt",
  exp_design = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/ed1.txt"
)

## Impute missing values in the data frame using the default minProb
## method prior to normalization.
imp_df <- impute_na(raw_df)

## Normalize the imp_df object using the default quantile method
norm_df1 <- normalize_data(imp_df)

## Use the cyclicloess method
norm_df2 <- normalize_data(imp_df, method = "cyclicloess")

## Normalize data in the raw_df object prior to imputation.
norm_df3 <- normalize_data(raw_df)
```

norm_plot

*Visualize the effect of normalization***Description**

This function visualizes the impact of normalization on the data

Usage

```
norm_plot(
  original,
  normalized,
  type = "box",
  text_size = 10,
  palette = "viridis",
  save = FALSE,
  file_path = NULL,
  file_name = "Norm_plot",
  file_type = "pdf",
  dpi = 80,
  plot_width = 10,
  plot_height = 7
)
```

Arguments

original	A raw_df object (output of create_df) containing missing values, or an imp_df object after imputing the missing values with impute_na.
normalized	A norm_df object after normalizing the data frame provided as original using normalize_data.
type	Type of plot to generate. Choices are "box" or "density." Default is "box."
text_size	Text size for plot labels, axis labels etc. Default is 10.
palette	Viridis color palette option for plots. Default is "viridis". See viridis for available options.
save	Logical. If TRUE saves a copy of the plot in the directory provided in file_path.
file_path	A string containing the directory path to save the file.
file_name	File name to save the plot. Default is "Norm_plot."
file_type	File type to save the plot. Default is "pdf".
dpi	Plot resolution. Default is 80.
plot_width	Width of the plot. Default is 10.
plot_height	Height of the plot. Default is 7.

Details

Given two data frames, one with data prior to normalization (original), and the other, after normalization (normalized), `norm_plot` generates side-by-side plots to visualize the effect of normalization on the protein intensity data.

Value

A `ggplot2` plot object.

Author(s)

Chathurani Ranathunge

See Also

- [normalize_data](#)
- `create_df`
- `impute_na`

Examples

```
## Generate a raw_df object with default settings. No technical replicates.
raw_df <- create_df(
  prot_groups = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/pg1.txt",
  exp_design = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/ed1.txt"
)

## Impute missing values in the data frame using the default minProb
## method.
imp_df <- impute_na(raw_df)

## Normalize the imp_df object using the default quantile method
norm_df <- normalize_data(imp_df)

## Visualize normalization using box plots
norm_plot(original = imp_df, normalized = norm_df)

## Visualize normalization using density plots
norm_plot(imp_df, norm_df, type = "density")
```

onegroup_only

Proteins that are only expressed in a given group

Description

This function outputs a list of proteins that are only expressed (present) in one user-specified group while not expressed (completely absent) in another user-specified group.

Usage

```
onegroup_only(
  raw_df,
  abs_group,
  pres_group,
  set_na = 0.34,
  save = FALSE,
  file_path = NULL
)
```

Arguments

<code>raw_df</code>	A <code>raw_df</code> object (output of <code>create_df</code>)
<code>abs_group</code>	Name of the group in which proteins are not expressed.
<code>pres_group</code>	Name of the group in which proteins are expressed.
<code>set_na</code>	The percentage of missing data allowed in <code>pres_group</code> . Default is 0.34 (one third of the samples in the group).
<code>save</code>	Logical. If TRUE (default), it saves the output in a text file named "Group_pres_group_only.txt."
<code>file_path</code>	A string containing the directory path to save the file.

Details

Note: `onegroup_only` function assumes that column names in the `raw_df` object provided as `df` follow "Group_UniqueSampleID" notation. (Use `head(raw_df)` to check the structure of your `raw_df` object.)

- Given a pair of groups, `onegroup_only` function finds proteins that are only expressed in `pres_group` while completely absent or not expressed in `abs_group`.
- A text file containing majority protein IDs will be saved in a temporary directory if `file_path` is not specified.

Value

A list of majority protein IDs.

Author(s)

Chathurani Ranathunge

Examples

```
# Generate a raw_df object with default settings. No technical replicates.
raw_df <- create_df(
  prot_groups = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/pg1.txt",
  exp_design = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/ed1.txt"
)

## Find the proteins only expressed in group L, but absent in group H.
```

```
onegroup_only(raw_df, abs_group = "H",
pres_group = "L")
```

performance_plot	<i>Model performance plot</i>
------------------	-------------------------------

Description

This function generates plots to visualize model performance

Usage

```
performance_plot(
  model_list,
  type = "box",
  text_size = 10,
  palette = "viridis",
  save = FALSE,
  file_path = NULL,
  file_name = "Performance_plot",
  file_type = "pdf",
  plot_width = 7,
  plot_height = 7,
  dpi = 80
)
```

Arguments

model_list	A model_list object from performing train_models.
type	Type of plot to generate. Choices are "box" or "dot." Default is "box." for boxplots.
text_size	Text size for plot labels, axis labels etc. Default is 10.
palette	Viridis color palette option for plots. Default is "viridis". See viridis for available options.
save	Logical. If TRUE saves a copy of the plot in the directory provided in file_path.
file_path	A string containing the directory path to save the file.
file_name	File name to save the plot. Default is "Performance_plot."
file_type	File type to save the plot. Default is "pdf".
plot_width	Width of the plot. Default is 7.
plot_height	Height of the plot. Default is 7.
dpi	Plot resolution. Default is 80.

Details

- performance_plot uses resampling results from models included in the model_list to generate plots showing model performance.
- The default metrics used for classification based models are "Accuracy" and "Kappa."
- These metric types can be changed by providing additional arguments to the train_models function. See [train](#) and [trainControl](#) for more information.

Value

A ggplot2 object.

Author(s)

Chathurani Ranathunge

See Also

- [train_models](#)
- [resamples](#)
- [train](#)
- [trainControl](#)

Examples

```
## Create a model_df object
covid_model_df <- pre_process(covid_fit_df, covid_norm_df)

## Split the data frame into training and test data sets
covid_split_df <- split_data(covid_model_df)

## Fit models based on the default list of machine learning (ML) algorithms
covid_model_list <- train_models(covid_split_df)

## Generate box plots to visualize performance of different ML algorithms
performance_plot(covid_model_list)

## Generate dot plots
performance_plot(covid_model_list, type = "dot")

## Change color palette
performance_plot(covid_model_list, type = "dot", palette = "inferno")
```

pre_process

*Pre-process protein intensity data for modeling***Description**

This function pre-processes protein intensity data from the top differentially expressed proteins identified with `find_dep` for modeling.

Usage

```
pre_process(
  fit_df,
  norm_df,
  sig = "adjP",
  sig_cutoff = 0.05,
  fc = 1,
  n_top = 20,
  find_highcorr = TRUE,
  corr_cutoff = 0.9,
  save_corrmatrix = FALSE,
  file_path = NULL,
  rem_highcorr = TRUE
)
```

Arguments

<code>fit_df</code>	A <code>fit_df</code> object from performing <code>find_dep</code> .
<code>norm_df</code>	The <code>norm_df</code> object from which the <code>fit_df</code> object was obtained.
<code>sig</code>	Criteria to denote significance in differential expression. Choices are "adjP" (default) for adjusted p-value or "P" for p-value.
<code>sig_cutoff</code>	Cutoff value for p-values and adjusted p-values in differential expression. Default is 0.05.
<code>fc</code>	Minimum absolute log-fold change to use as threshold for differential expression. Default is 1.
<code>n_top</code>	The number of top hits from <code>find_dep</code> to be used in modeling. Default is 20.
<code>find_highcorr</code>	Logical. If TRUE (default), finds highly correlated proteins.
<code>corr_cutoff</code>	A numeric value specifying the correlation cutoff. Default is 0.90.
<code>save_corrmatrix</code>	Logical. If TRUE, saves a copy of the protein correlation matrix in a tab-delimited text file labeled "Protein_correlation.txt" in the directory specified by <code>file_path</code> .
<code>file_path</code>	A string containing the directory path to save the file.
<code>rem_highcorr</code>	Logical. If TRUE (default), removes highly correlated proteins (predictors or features).

Details

This function creates a data frame that contains protein intensities for a user-specified number of top differentially expressed proteins.

- Using `find_highcorr = TRUE`, highly correlated proteins can be identified, and can be removed with `rem_highcorr = TRUE`.
- Note: Most models will benefit from reducing correlation between proteins (predictors or features), therefore we recommend removing those proteins at this stage to reduce pairwise-correlation.
- If no or few proteins meet the significance threshold for differential expression, you may adjust `sig`, `fc`, and/or `sig_cutoff` accordingly to obtain more proteins for modeling.

Value

A `model_df` object, which is a data frame of protein intensities with proteins indicated by columns.

Author(s)

Chathurani Ranathunge

See Also

- `find_dep`, `normalize_data`
- [caret: findCorrelation](#)

Examples

```
## Create a model_df object with default settings.
covid_model_df1 <- pre_process(fit_df = covid_fit_df, norm_df = covid_norm_df)

## Change the correlation cutoff.
covid_model_df2 <- pre_process(covid_fit_df, covid_norm_df, corr_cutoff = 0.95)

## Change the significance criteria to include more proteins
covid_model_df3 <- pre_process(covid_fit_df, covid_norm_df, sig = "P")

## Change the number of top differentially expressed proteins to include
covid_model_df4 <- pre_process(covid_fit_df, covid_norm_df, sig = "P", n_top = 24)
```

rem_feature

Remove user-specified proteins (features) from a data frame

Description

This function removes user-specified proteins from a `model_df` object

Usage

```
rem_feature(model_df, rem_protein)
```

Arguments

model_df	A model_df object.
rem_protein	Name of the protein to remove.

Details

- After visualizing protein intensity variation among conditions with `feature_plot` or after assessing the importance of each protein in models using `varimp_plot`, you can choose to remove specific proteins (features) from the data frame.
- For example, you can choose to remove a protein from the `model_df` object if the protein does not show distinct patterns of variation among conditions. This protein may show mostly overlapping distributions in the feature plots.
- Another incidence would be removing a protein that is very low in variable importance in the models built using `train_models`. You can visualize variable importance using `varimp_plot`.

Value

A model_df object.

Author(s)

Chathurani Ranathunge

See Also

[feature_plot](#), [pre_process](#)

Examples

```
covid_model_df <- pre_process(fit_df = covid_fit_df, norm_df = covid_norm_df)

## Remove sp|P22352|GPX3_HUMAN protein from the model_df object
covid_model_df1 <- rem_feature(covid_model_df, rem_protein = "sp|P22352|GPX3_HUMAN")
```

rem_sample	<i>Remove user-specified samples</i>
------------	--------------------------------------

Description

This function removes user-specified samples from the data frame.

Usage

```
rem_sample(raw_df, rem)
```

Arguments

raw_df	A raw_df object.
rem	Name of the sample to remove.

Details

- rem_sample assumes that sample names follow the "Group_UniqueSampleID_TechnicalReplicate" notation (Use head(raw_df) to see the structure of the raw_df object.)
- If all the technical replicates representing a sample needs to be removed, provide "Group_UniqueSampleID" as rem.
- If a specific technical replicate needs to be removed in case it shows weak correlation with other technical replicates for example, you can remove that particular technical replicate by providing "Group_UniqueSampleID_TechnicalReplicate" as rem.

Value

A raw_df object.

Author(s)

Chathurani Ranathunge

See Also

[corr_plot](#), [create_df](#)

Examples

```
## Use a data set containing technical replicates to create a raw_df object
raw_df <- create_df(
  prot_groups = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/pg2.txt",
  exp_design = "https://raw.githubusercontent.com/caranathunge/promor_example_data/main/ed2.txt",
  tech_reps = TRUE
)
# Check the first few rows of the raw_df object
head(raw_df)
```

```
## Remove all technical replicates of "WT_4"
raw_df1 <- rem_sample(raw_df, "WT_4")

## Remove only technical replicate number 2 of "WT_4"
raw_df2 <- rem_sample(raw_df, "WT_4_2")
```

roc_plot

ROC plot

Description

This function generates Receiver Operating Characteristic (ROC) curves to evaluate models

Usage

```
roc_plot(
  probability_list,
  split_df,
  ...,
  multiple_plots = TRUE,
  text_size = 10,
  palette = "viridis",
  save = FALSE,
  file_path = NULL,
  file_name = "ROC_plot",
  file_type = "pdf",
  plot_width = 7,
  plot_height = 7,
  dpi = 80
)
```

Arguments

probability_list	A probability_list object from performing test_models with type = "prob".
split_df	A split_df object from performing split_data
...	Additional arguments to be passed on to roc .
multiple_plots	Logical. If FALSE plots all ROC curves representing algorithms included in the probability_list in a single plot.
text_size	Text size for plot labels, axis labels etc. Default is 10.
palette	Viridis color palette option for plots. Default is "viridis". See viridis for available options.
save	Logical. If TRUE saves a copy of the plot in the directory provided in file_path.
file_path	A string containing the directory path to save the file.

file_name	File name to save the plot. Default is "ROC_plot."
file_type	File type to save the plot. Default is "pdf".
plot_width	Width of the plot. Default is 7.
plot_height	Height of the plot. Default is 7.
dpi	Plot resolution. Default is 80.

Details

- roc_plot first uses probabilities generated during test_models to build a ROC object.
- Next, relevant information is extracted from the ROC object to plot the ROC curves.

Value

A ggplot2 object.

Author(s)

Chathurani Ranathunge

See Also

- test_models
- [roc](#)

Examples

```
## Create a model_df object
covid_model_df <- pre_process(covid_fit_df, covid_norm_df)

## Split the data frame into training and test data sets
covid_split_df <- split_data(covid_model_df)

## Fit models using the default list of machine learning (ML) algorithms
covid_model_list <- train_models(covid_split_df)

# Test a list of models on a test data set and output class probabilities,
covid_prob_list <- test_models(covid_model_list, covid_split_df, type = "prob")

## Plot ROC curves separately for each ML algorithm
roc_plot(covid_prob_list, covid_split_df)

## Plot all ROC curves in one plot
roc_plot(covid_prob_list, covid_split_df, multiple_plots = FALSE)

## Change color palette
roc_plot(covid_prob_list, covid_split_df, palette = "plasma")
```

split_data	<i>Split the data frame to create training and test data</i>
------------	--

Description

This function can be used to create balanced splits of the protein intensity data in a `model_df` object to create training and test data

Usage

```
split_data(model_df, train_size = 0.8, seed = NULL)
```

Arguments

<code>model_df</code>	A <code>model_df</code> object from performing <code>pre_process</code> .
<code>train_size</code>	The size of the training data set as a proportion of the complete data set. Default is 0.8.
<code>seed</code>	Numerical. Random number seed. Default is <code>NULL</code>

Details

This function splits the `model_df` object in to training and test data sets using random sampling while preserving the original class distribution of the data. Make sure to fix the random number seed with `seed` for reproducibility

Value

A list of data frames.

Author(s)

Chathurani Ranathunge

See Also

- `pre_process`
- [createDataPartition](#)

Examples

```
## Create a model_df object
covid_model_df <- pre_process(covid_fit_df, covid_norm_df)

## Split the data frame into training and test data sets using default settings
covid_split_df1 <- split_data(covid_model_df, seed = 8314)

## Split the data frame into training and test data sets with 70% of the
## data in training and 30% in test data sets
```

```
covid_split_df2 <- split_data(covid_model_df, train_size = 0.7, seed = 8314)

## Access training data set
covid_split_df1$training

## Access test data set
covid_split_df1$test
```

test_models

Test machine learning models on test data

Description

This function can be used to predict test data using models generated by different machine learning algorithms

Usage

```
test_models(
  model_list,
  split_df,
  type = "prob",
  save_confusionmatrix = FALSE,
  file_path = NULL,
  ...
)
```

Arguments

model_list	A model_list object from performing train_models.
split_df	A split_df object from performing split_data.
type	Type of output. Set type as "prob" (default) to output class probabilities, and "raw" to output class predictions.
save_confusionmatrix	Logical. If TRUE, a tab-delimited text file ("Confusion_matrices.txt") with confusion matrices in the long-form data format will be saved in the directory specified by file_path. See below for more details.
file_path	A string containing the directory path to save the file.
...	Additional arguments to be passed on to predict .

Details

- test_models function uses models obtained from train_models to predict a given test data set.
- Setting type = "raw" is required to obtain confusion matrices.
- Setting type = "prob" (default) will output a list of probabilities that can be used to generate ROC curves using roc_plot.

Value

- `probability_list`: If `type = "prob"`, a list of data frames containing class probabilities for each method in the `model_list` will be returned.
- `prediction_list`: If `type = "raw"`, a list of factors containing class predictions for each method will be returned.

Author(s)

Chathurani Ranathunge

See Also

- `split_df`
- `train_models`
- [predict](#)
- [confusionMatrix](#)

Examples

```
## Create a model_df object
covid_model_df <- pre_process(covid_fit_df, covid_norm_df)

## Split the data frame into training and test data sets
covid_split_df <- split_data(covid_model_df)

## Fit models using the default list of machine learning (ML) algorithms
covid_model_list <- train_models(covid_split_df)

# Test a list of models on a test data set and output class probabilities,
covid_prob_list <- test_models(model_list = covid_model_list, split_df = covid_split_df)

## Not run:
# Save confusion matrices in the working directory and output class predictions
covid_pred_list <- test_models(
  model_list = covid_model_list,
  split_df = covid_split_df,
  type = "raw",
  save_confusionmatrix = TRUE,
  file_path = "."
)

## End(Not run)
```

train_models	<i>Train machine learning models on training data</i>
--------------	---

Description

This function can be used to train models on protein intensity data using different machine learning algorithms

Usage

```
train_models(
  split_df,
  resample_method = "repeatedcv",
  resample_iterations = 10,
  num_repeats = 3,
  algorithm_list,
  seed = NULL,
  ...
)
```

Arguments

split_df	A split_df object from performing split_data.
resample_method	The resampling method to use. Default is "repeatedcv" for repeated cross validation. See trainControl for details on other available methods.
resample_iterations	Number of resampling iterations. Default is 10.
num_repeats	The number of complete sets of folds to compute (For resampling method = "repeatedcv" only).
algorithm_list	A list of classification or regression algorithms to use. A full list of machine learning algorithms available through the caret package can be found here: http://topepo.github.io/caret/train-models-by-tag.html . See below for default options.
seed	Numerical. Random number seed. Default is NULL
...	Additional arguments to be passed on to train function in the caret package.

Details

- train_models function can be used to first define the control parameters to be used in training models, calculate resampling-based performance measures for models based on a given set of machine-learning algorithms, and output the best model for each algorithm.
- In the event that algorithm_list is not provided, a default list of four classification-based machine-learning algorithms will be used for building and training models. Default algorithm_list: "svmRadial", "rf", "glm", "xgbLinear, and "naive_bayes."
- Note: Models that fail to build are removed from the output.
- Make sure to fix the random number seed with seed for reproducibility

Value

A list of class `train` for each machine-learning algorithm. See [train](#) for more information on accessing different elements of this list.

Author(s)

Chathurani Ranathunge

References

Kuhn, Max. "Building predictive models in R using the caret package." Journal of statistical software 28 (2008): 1-26.

See Also

- `pre_process`
- [trainControl](#)
- [train](#)

Examples

```
## Create a model_df object
covid_model_df <- pre_process(covid_fit_df, covid_norm_df)

## Split the data frame into training and test data sets
covid_split_df <- split_data(covid_model_df, seed = 8314)

## Fit models based on the default list of machine learning (ML) algorithms
covid_model_list1 <- train_models(split_df = covid_split_df, seed = 351)

## Fit models using a user-specified list of ML algorithms.
covid_model_list2 <- train_models(
  covid_split_df,
  algorithm_list = c("svmRadial", "glmboost"),
  seed = 351
)

## Change resampling method and resampling iterations.
covid_model_list3 <- train_models(
  covid_split_df,
  resample_method = "cv",
  resample_iterations = 50,
  seed = 351
)
```

varimp_plot	<i>Variable importance plot</i>
-------------	---------------------------------

Description

This function visualizes variable importance in models

Usage

```
varimp_plot(
  model_list,
  ...,
  type = "lollipop",
  text_size = 10,
  palette = "viridis",
  n_row,
  n_col,
  save = FALSE,
  file_path = NULL,
  file_name = "VarImp_plot",
  file_type = "pdf",
  dpi = 80,
  plot_width = 7,
  plot_height = 7
)
```

Arguments

model_list	A model_list object from performing train_models.
...	Additional arguments to be passed on to varImp .
type	Type of plot to generate. Choices are "bar" or "lollipop." Default is "lollipop."
text_size	Text size for plot labels, axis labels etc. Default is 10.
palette	Viridis color palette option for plots. Default is "viridis". See viridis for available options.
n_row	Number of rows to print the plots.
n_col	Number of columns to print the plots.
save	Logical. If TRUE saves a copy of the plot in the directory provided in file_path.
file_path	A string containing the directory path to save the file.
file_name	File name to save the plot. Default is "VarImp_plot."
file_type	File type to save the plot. Default is "pdf".
dpi	Plot resolution. Default is 80.
plot_width	Width of the plot. Default is 7.
plot_height	Height of the plot. Default is 7.

Details

- `varimp_plot` produces a list of plots showing variable importance measures calculated from models generated with different machine-learning algorithms.
- Note: Variables are ordered by variable importance in descending order, and by default, importance values are scaled to 0 and 100. This can be changed by specifying `scale = FALSE`. See [varImp](#) for more information.

Value

A list of `ggplot2` objects.

Author(s)

Chathurani Ranathunge

See Also

- `train_models`, `rem_feature`
- [varImp](#)

Examples

```
## Create a model_df object
covid_model_df <- pre_process(covid_fit_df, covid_norm_df)

## Split the data frame into training and test data sets
covid_split_df <- split_data(covid_model_df)

## Fit models based on the default list of machine learning (ML) algorithms
covid_model_list <- train_models(covid_split_df)

## Variable importance - lollipop plots
varimp_plot(covid_model_list)

## Bar plots
varimp_plot(covid_model_list, type = "bar")

## Do not scale variable importance values
varimp_plot(covid_model_list, scale = FALSE)

## Change color palette
varimp_plot(covid_model_list, palette = "magma")
```

volcano_plot

*Volcano plot***Description**

This function generates volcano plots to visualize differentially expressed proteins between groups.

Usage

```
volcano_plot(
  fit_df,
  adj_method = "BH",
  sig = "adjP",
  cutoff = 0.05,
  lfc = 1,
  line_lfc = TRUE,
  line_p = TRUE,
  palette = "viridis",
  text_size = 10,
  label_top = FALSE,
  n_top = 10,
  save = FALSE,
  file_path = NULL,
  file_name = "Volcano_plot",
  file_type = "pdf",
  plot_height = 7,
  plot_width = 7,
  dpi = 80
)
```

Arguments

<code>fit_df</code>	A <code>fit_df</code> object from performing <code>find_dep</code> .
<code>adj_method</code>	Method used for adjusting the p-values for multiple testing. Default is "BH".
<code>sig</code>	Criteria to denote significance. Choices are "adjP" (default) for adjusted p-value or "P" for p-value.
<code>cutoff</code>	Cutoff value for p-values and adjusted p-values. Default is 0.05.
<code>lfc</code>	Minimum absolute log2-fold change to use as threshold for differential expression.
<code>line_lfc</code>	Logical. If TRUE(default), a dotted line will be shown to indicate the lfc threshold in the plot.
<code>line_p</code>	Logical. If TRUE(default), a dotted line will be shown to indicate the p-value or adjusted p-value cutoff.
<code>palette</code>	Viridis color palette option for plots. Default is "viridis". See viridis for available options.

text_size	Text size for axis text, labels etc.
label_top	Logical. If TRUE (default), labels are added to the dots to indicate protein names.
n_top	The number of top hits to label with protein name when label_top = TRUE. Default is 10.
save	Logical. If TRUE saves a copy of the plot in the directory provided in file_path.
file_path	A string containing the directory path to save the file.
file_name	File name to save the plot. Default is "Volcano_plot."
file_type	File type to save the plot. Default is "pdf".
plot_height	Height of the plot. Default is 7.
plot_width	Width of the plot. Default is 7.
dpi	Plot resolution. Default is 80.

Details

- Volcano plots show log-2-fold change on the x-axis, and based on the significance criteria chosen, either -log10(p-value) or -log10(adjusted p-value) on the y-axis.
- volcano_plot requires a fit_df object from performing differential expression analysis with find_dep.
- User has the option to choose criteria that denote significance.

Value

A ggplot2 plot object.

Author(s)

Chathurani Ranathunge

See Also

- find_dep
- topTable and lmFit functions from the limma package.

Examples

```
## Create a volcano plot with default settings.
volcano_plot(ecoli_fit_df)

## Change significance criteria and cutoff
volcano_plot(ecoli_fit_df, cutoff = 0.1, sig = "P")

## Label top 30 differentially expressed proteins and
## change the color palette of the plot
volcano_plot(ecoli_fit_df, label_top = TRUE, n_top = 30, palette = "mako")
```

Index

* datasets

- covid_fit_df, [5](#)
- covid_norm_df, [5](#)
- ecoli_fit_df, [8](#)
- ecoli_norm_df, [9](#)

avearrays, [3](#)

aver_techreps, [2](#)

confusionMatrix, [37](#)

corr_plot, [3](#), [32](#)

covid_fit_df, [5](#)

covid_norm_df, [5](#)

create_df, [3](#), [4](#), [6](#), [11](#), [12](#), [16–19](#), [21](#), [24](#), [26](#), [32](#)

createDataPartition, [35](#)

eBayes, [13](#)

ecoli_fit_df, [8](#)

ecoli_norm_df, [9](#)

feature_plot, [9](#), [31](#)

filterbygroup_na, [11](#)

find_dep, [12](#)

heatmap_de, [14](#)

heatmap_na, [16](#)

impute_na, [18](#)

impute_plot, [20](#)

kNN, [19](#)

limma, [3](#), [13](#), [15](#), [43](#)

lmFit, [13](#), [15](#), [43](#)

missForest, [19](#)

norm_plot, [24](#)

normalize_data, [22](#), [25](#)

normalizeBetweenArrays, [23](#)

onengroup_only, [25](#)

pca, [19](#)

pcaMethods, [19](#)

performance_plot, [27](#)

pre_process, [29](#), [31](#)

predict, [36](#), [37](#)

rem_feature, [30](#)

rem_sample, [32](#)

resamples, [28](#)

roc, [33](#), [34](#)

roc_plot, [33](#)

split_data, [35](#)

test_models, [36](#)

topTable, [13](#), [15](#), [43](#)

train, [28](#), [38](#), [39](#)

train_models, [38](#)

trainControl, [28](#), [38](#), [39](#)

varImp, [40](#), [41](#)

varimp_plot, [40](#)

VIM, [19](#)

viridis, [4](#), [10](#), [15](#), [17](#), [21](#), [24](#), [27](#), [33](#), [40](#), [42](#)

volcano_plot, [42](#)

write.fit, [13](#)