Package 'PrInCE'

November 21, 2024

Title Predicting Interactomes from Co-Elution

Version 1.23.0

BugReports https://github.com/fosterlab/PrInCE/issues

Description PrInCE (Predicting Interactomes from Co-Elution) uses a naive Bayes classifier trained on dataset-derived features to recover protein-protein interactions from co-elution chromatogram profiles. This package contains the R implementation of PrInCE.

Depends R (>= 3.6.0)

Imports purrr (>= 0.2.4), dplyr (>= 0.7.4), tidyr (>= 0.8.99), forecast (>= 8.2), progress (>= 1.1.2), Hmisc (>= 4.0), naivebayes (>= 0.9.1), robustbase (>= 0.92-7), ranger (>= 0.8.0), LiblineaR (>= 2.10-8), speedglm (>= 0.3-2), tester (>= 0.1.7), magrittr (>= 1.5), Biobase (>= 2.40.0), MSnbase (>= 2.8.3), stats, utils, methods, Rdpack (>= 0.7)

Suggests BiocStyle, knitr, rmarkdown

biocViews Proteomics, SystemsBiology, NetworkInference

VignetteBuilder knitr

License GPL-3 + file LICENSE

Encoding UTF-8

RdMacros Rdpack

RoxygenNote 7.1.1

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

git_branch devel

git_last_commit 92c3d03

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2024-11-20

Author Michael Skinnider [aut, trl, cre],
R. Greg Stacey [ctb],
Nichollas Scott [ctb],

Anders Kristensen [ctb],
Leonard Foster [aut, led]

Maintainer Michael Skinnider <michael.skinnider@msl.ubc.ca>

Contents

adjacency_matrix_from_data_frame

Create an adjacency matrix from a data frame

Description

Convert a data frame containing pairwise interactions into an adjacency matrix. The resulting square adjacency matrix contains ones for proteins that are found in interactions and zeroes otherwise.

Usage

```
adjacency_matrix_from_data_frame(dat, symmetric = TRUE, node_columns = c(1, 2))
```

Arguments

dat	a data frame containing pairwise interactions
symmetric	if true, interactions in both directions will be added to the adjacency matrix
node_columns	a vector of length two, denoting either the indices (integer vector) or column names (character vector) of the columns within the data frame containing the nodes participating in pairwise interactions; defaults to the first two columns of the data frame (c(1, 2))

Value

an adjacency matrix between all interacting proteins

Examples

```
ppi <- data.frame(protein_A = paste0("protein", seq_len(10)),
                  protein_B = paste0("protein", c(rep(3, 2), rep(5, 5),
                                      rep(7, 3))))
adj <- adjacency_matrix_from_data_frame(ppi)
```

adjacency_matrix_from_list

Create an adjacency matrix from a list of complexes

Description

Convert a list of complexes into a pairwise adjacency matrix. The resulting square adjacency matrix contains ones for proteins that are found in the same complex and zeroes otherwise.

Usage

```
adjacency_matrix_from_list(complexes)
```

Arguments

complexes	a list of complexes, with each entry containing complex subunits as a character vector

Value

an adjacency matrix between all complex subunits

Examples

```
data(gold_standard)
adj <- adjacency_matrix_from_list(gold_standard)
```

---

aic *Model selection for Gaussian mixture models*

Description

Calculate the AIC, corrected AIC, or BIC for a curve fit with a Gaussian mixture model by non-linear least squares optimization. This function permits the calculation of the AIC/AICc/BIC after rejecting some Gaussians in the model, for example because their centres are outside the bounds of the profile.

Usage

```
gaussian_aic(coefs, chromatogram)

gaussian_aicc(coefs, chromatogram)

gaussian_bic(coefs, chromatogram)
```

Arguments

coefs	the coefficients of the Gaussian mixture model, output by fit_gaussians
chromatogram	the raw elution profile

Value

the AIC, corrected AIC, or BIC of the fit model

build_gaussians *Deconvolve profiles into Gaussian mixture models*

Description

Identify peaks in co-fractionation profiles by deconvolving peaks in Gaussian mixture models. Models are mixtures of between 1 and 5 Gaussians. Profiles are pre-processed prior to building Gaussians by filtering and cleaning. By default, profiles with fewer than 5 non-missing points, or fewer than 5 consecutive points after imputation of single missing values, are removed. Profiles are cleaned by replacing missing values with near-zero noise, imputing single missing values as the mean of neighboring points, and smoothing with a moving average filter.

Usage

```
build_gaussians(
  profile_matrix,
  min_points = 1,
  min_consecutive = 5,
  impute_NA = TRUE,
  smooth = TRUE,
  smooth_width = 4,
  max_gaussians = 5,
  criterion = c("AICc", "AIC", "BIC"),
  max_iterations = 50,
  min_R_squared = 0.5,
  method = c("guess", "random"),
  filter_gaussians_center = TRUE,
  filter_gaussians_height = 0.15,
  filter_gaussians_variance_min = 0.5,
  filter_gaussians_variance_max = 50
)
```

Arguments

profile_matrix	a numeric matrix of co-elution profiles, with proteins in rows, or a MSnSet object
min_points	filter profiles without at least this many total, non-missing points; passed to filter_profiles
min_consecutive	filter profiles without at least this many consecutive, non-missing points; passed to filter_profiles
impute_NA	if true, impute single missing values with the average of neighboring values; passed to clean_profiles
smooth	if true, smooth the chromatogram with a moving average filter; passed to clean_profiles
smooth_width	width of the moving average filter, in fractions; passed to clean_profiles

`max_gaussians` the maximum number of Gaussians to fit; defaults to 5. Note that Gaussian mixtures with more parameters than observed (i.e., non-zero or NA) points will not be fit. Passed to `choose_gaussians`

`criterion` the criterion to use for model selection; one of "AICc" (corrected AIC, and default), "AIC", or "BIC". Passed to `choose_gaussians`

`max_iterations` the number of times to try fitting the curve with different initial conditions; defaults to 50. Passed to `fit_gaussians`

`min_R_squared` the minimum R-squared value to accept when fitting the curve with different initial conditions; defaults to 0.5. Passed to `fit_gaussians`

`method` the method used to select the initial conditions for nonlinear least squares optimization (one of "guess" or "random"); see `make_initial_conditions` for details. Passed to `fit_gaussians`

`filter_gaussians_center`
: true or false: filter Gaussians whose centres fall outside the bounds of the chromatogram. Passed to `fit_gaussians`

`filter_gaussians_height`
: Gaussians whose heights are below this fraction of the chromatogram height will be filtered. Setting this value to zero disables height-based filtering of fit Gaussians. Passed to `fit_gaussians`

`filter_gaussians_variance_min`
: Gaussians whose variance falls below this number of fractions will be filtered. Setting this value to zero disables filtering. Passed to `fit_gaussians`

`filter_gaussians_variance_max`
: Gaussians whose variance is above this number of fractions will be filtered. Setting this value to zero disables filtering. Passed to `fit_gaussians`

Value

a list of fit Gaussian mixture models, where each item in the list contains the following five fields: the number of Gaussians used to fit the curve; the R^2 of the fit; the number of iterations used to fit the curve with different initial conditions; the coefficients of the fit model; and the curve predicted by the fit model. Profiles that could not be fit by a Gaussian mixture model above the minimum R-squared cutoff will be absent from the returned list.

Examples

```
data(scott)
mat <- clean_profiles(scott[seq_len(5), ])
gauss <- build_gaussians(mat, max_gaussians = 3)
```

`calculate_autocorrelation` *Calculate the autocorrelation for each protein between a pair of co-elution experiments.*

Description

For a given protein, the correlation coefficient to all other proteins in the first condition is calculated, yielding a vector of correlation coefficients. The same procedure is repeated for the second condition, and the two vectors of correlation coefficients are themselves correlated, yielding a metric whereby higher values reflect proteins with unchanging interaction profiles between conditions, while lower values reflect proteins with substantially changing interaction profiles.

Usage

```
calculate_autocorrelation(
  profile1,
  profile2,
  cor_method = c("pearson", "spearman", "kendall"),
  min_replicates = 1,
  min_fractions = 1,
  min_pairs = 0
)
```

Arguments

`profile1`	a numeric matrix or data frame with proteins in rows and fractions in columns, or a `MSnSet` object, representing the first co-elution condition
`profile2`	a numeric matrix or data frame with proteins in rows and fractions in columns, or a `MSnSet` object, representing the second co-elution condition
`cor_method`	the correlation method to use; one of `"pearson"`, `"spearman"`, or `"kendall"`).
`min_fractions`	filter proteins not quantified in at least this many fractions
`min_pairs`	remove correlations between protein pairs not co-occuring in at least this many fractions from the autocorrelation calculation

Details

Note that all of zero, `NA`, `NaN`, and infinite values are all treated equivalently as missing values when applying the `min_fractions` and `min_pairs` filters, but different handling of missing values will produce different autocorrelation scores.

Value

a named vector of autocorrelation scores for all proteins found in both matrices.

`calculate_features` *Calculate the default features used to predict interactions in PrInCE*

Description

Calculate the six features that are used to discriminate interacting and non-interacting protein pairs based on co-elution profiles in PrInCE, namely: raw Pearson R value, cleaned Pearson R value, raw Pearson P-value, Euclidean distance, co-peak, and co-apex. Optionally, one or more of these can be disabled.

Usage

```
calculate_features(
  profile_matrix,
  gaussians,
  min_pairs = 0,
  pearson_R_raw = TRUE,
  pearson_R_cleaned = TRUE,
  pearson_P = TRUE,
  euclidean_distance = TRUE,
  co_peak = TRUE,
  co_apex = TRUE,
  n_pairs = FALSE,
  max_euclidean_quantile = 0.9
)
```

Arguments

`profile_matrix`	a numeric matrix of co-elution profiles, with proteins in rows, or a `MSnSet` object
`gaussians`	a list of Gaussian mixture models fit to the profile matrix by `link{build_gaussians}`
`min_pairs`	minimum number of overlapping fractions between any given protein pair to consider a potential interaction
`pearson_R_raw`	if true, include the Pearson correlation (R) between raw profiles as a feature
`pearson_R_cleaned`	if true, include the Pearson correlation (R) between cleaned profiles as a feature
`pearson_P`	if true, include the P-value of the Pearson correlation between raw profiles as a feature
`euclidean_distance`	if true, include the Euclidean distance between cleaned profiles as a feature
`co_peak`	if true, include the 'co-peak score' (that is, the distance, in fractions, between the single highest value of each profile) as a feature
`co_apex`	if true, include the 'co-apex score' (that is, the minimum Euclidean distance between any pair of fit Gaussians) as a feature

max_euclidean_quantile
: very high Euclidean distance values are trimmed to avoid numerical precision issues; values above this quantile will be replaced with the value at this quantile (default: 0.9)

Value

a data frame containing the calculated features for all possible protein pairs

---

calculate_precision *Calculate precision at each point in a sequence*

Description

Calculate the precision of a list of interactions at each point in the list, given a set of labels.

Usage

```
calculate_precision(labels)
```

Arguments

labels	a vector of zeroes (FPs) and ones (TPs)

Value

a vector of the same length giving the precision at each point in the input vector

Examples

```
## calculate features
data(scott)
data(scott_gaussians)
subset <- scott[seq_len(500), ] ## limit to first 500 proteins
gauss <- scott_gaussians[names(scott_gaussians) %in% rownames(subset)]
features <- calculate_features(subset, gauss)
## make training labels
data(gold_standard)
ref <- adjacency_matrix_from_list(gold_standard)
labels <- make_labels(ref, features)
## predict interactions with naive Bayes classifier
ppi <- predict_ensemble(features, labels, classifier = "NB", cv_folds = 3,
                        models = 1)
## tag precision of each interaction
ppi$precision <- calculate_precision(ppi$label)
```

check_gaussians *Check the format of a list of Gaussians*

Description

Test whether an input list of Gaussians conforms to the format expected by PrInCE: that is, a named list with five fields for each entry, i.e., the number of Gaussians in the mixture model, the r^2 value, the number of iterations used by `nls`, the coefficients of each model, and the fitted curve.

Usage

```
check_gaussians(
  gaussians,
  proteins = NULL,
  replicate_idx = NULL,
  n_error = 3,
  pct_warning = 0.1
)
```

Arguments

`gaussians`	the list of Gaussians
`proteins`	the complete set of input proteins
`replicate_idx`	the replicate being analyzed, if input proteins are provided; used to throw more informative error messages
`n_error`	minimum number of proteins that can have fitted Gaussians without throwing an error
`pct_warning`	minimum fraction of proteins that can have fitted Gaussians without giving a warning

Details

Optionally, some extra checks will be done on the fraction of proteins in the complete dataset for which a Gaussian mixture model could be fit, if provided. In particular, the function will throw an error if fewer than `n_error` proteins have a fitted Gaussian, and emit a warning if fewer than `pct_warning` do.

Value

`TRUE` if all conditions are met, but throws an error if any is not

Examples

```
data(scott_gaussians)
check_gaussians(scott_gaussians)
```

choose_gaussians *Fit a Gaussian mixture model to a co-elution profile*

Description

Fit mixtures of one or more Gaussians to the curve formed by a chromatogram profile, and choose the best fitting model using an information criterion of choice.

Usage

```
choose_gaussians(
  chromatogram,
  points = NULL,
  max_gaussians = 5,
  criterion = c("AICc", "AIC", "BIC"),
  max_iterations = 10,
  min_R_squared = 0.5,
  method = c("guess", "random"),
  filter_gaussians_center = TRUE,
  filter_gaussians_height = 0.15,
  filter_gaussians_variance_min = 0.1,
  filter_gaussians_variance_max = 50
)
```

Arguments

chromatogram	a numeric vector corresponding to the chromatogram trace
points	optional, the number of non-NA points in the raw data
max_gaussians	the maximum number of Gaussians to fit; defaults to 5. Note that Gaussian mixtures with more parameters than observed (i.e., non-zero or NA) points will not be fit.
criterion	the criterion to use for model selection; one of "AICc" (corrected AIC, and default), "AIC", or "BIC"
max_iterations	the number of times to try fitting the curve with different initial conditions; defaults to 10
min_R_squared	the minimum R-squared value to accept when fitting the curve with different initial conditions; defaults to 0.5
method	the method used to select the initial conditions for nonlinear least squares optimization (one of "guess" or "random"); see make_initial_conditions for details
filter_gaussians_center	true or false: filter Gaussians whose centres fall outside the bounds of the chromatogram

filter_gaussians_height
: Gaussians whose heights are below this fraction of the chromatogram height will be filtered. Setting this value to zero disables height-based filtering of fit Gaussians

filter_gaussians_variance_min
: Gaussians whose variance is below this threshold will be filtered. Setting this value to zero disables filtering.

filter_gaussians_variance_max
: Gaussians whose variance is above this threshold will be filtered. Setting this value to zero disables filtering.

Value

a list with five entries: the number of Gaussians used to fit the curve; the R^2 of the fit; the number of iterations used to fit the curve with different initial conditions; the coefficients of the fit model; and the curve predicted by the fit model.

Examples

```
data(scott)
chrom <- clean_profile(scott[1, ])
gauss <- choose_gaussians(chrom, max_gaussians = 3)
```

---

clean_profile *Preprocess a co-elution profile*

Description

Clean a co-elution/co-fractionation profile by (1) imputing single missing values with the average of neighboring values, (2) replacing missing values with random, near-zero noise, and (3) smoothing with a moving average filter.

Usage

```
clean_profile(
  chromatogram,
  impute_NA = TRUE,
  smooth = TRUE,
  smooth_width = 4,
  noise_floor = 0.001
)
```

Arguments

chromatogram	a numeric vector corresponding to the chromatogram trace
impute_NA	if true, impute single missing values with the average of neighboring values
smooth	if true, smooth the chromatogram with a moving average filter
smooth_width	width of the moving average filter, in fractions
noise_floor	mean value of the near-zero noise to add

Value

a cleaned profile

Examples

```
data(scott)
chrom <- scott[16, ]
cleaned <- clean_profile(chrom)
```

---

clean_profiles *Preprocess a co-elution profile matrix*

Description

Clean a matrix of co-elution/co-fractionation profiles by (1) imputing single missing values with the average of neighboring values, (2) replacing missing values with random, near-zero noise, and (3) smoothing with a moving average filter.

Usage

```
clean_profiles(
  profile_matrix,
  impute_NA = TRUE,
  smooth = TRUE,
  smooth_width = 4,
  noise_floor = 0.001
)
```

Arguments

profile_matrix	a numeric matrix of co-elution profiles, with proteins in rows, or a MSnSet object
impute_NA	if true, impute single missing values with the average of neighboring values
smooth	if true, smooth the chromatogram with a moving average filter
smooth_width	width of the moving average filter, in fractions
noise_floor	mean value of the near-zero noise to add

Value

a cleaned matrix

Examples

```
data(scott)
mat <- scott[c(1, 16), ]
mat_clean <- clean_profiles(mat)
```

`concatenate_features` *Combine features across multiple replicates*

Description

Concatenate features extracted from multiple replicates to a single data frame that will be used as input to a classifier. Doing so allows the classifier to naturally weight evidence for an interaction between each protein pair from each feature in each replicate in proportion to its discriminatory power on known examples.

Usage

```
concatenate_features(feature_list)
```

Arguments

`feature_list`	a list of feature data frames, as produced by `calculate_features`, with proteins in the first two columns

Value

a data frame containing features for all protein pairs across all replicates

---

`co_apex` *Calculate the co-apex score for every protein pair*

Description

Calculate the co-apex score for every pair of proteins. This is defined as the minimum Euclidean distance between any two Gaussians fit to each profile.

Usage

```
co_apex(gaussians, proteins = NULL)
```

Arguments

gaussians	a list of Gaussian mixture models fit to the profile matrix by link{build_gaussians}
proteins	all proteins being scored, optionally including those without Gaussian fits

Value

a matrix of co-apex scores

Examples

```
data(scott_gaussians)
gauss <- scott_gaussians[seq_len(25)]
CA <- co_apex(gauss)
```

---

detect_complexes *Detect significantly interacting complexes in a chromatogram matrix*

Description

Use a permutation testing approach to identify complexes that show a significant tendency to interact, relative to random sets of complexes of equivalent size. The function begins by calculating the Pearson correlation or Euclidean distance between all proteins in the matrix, and

Usage

```
detect_complexes(
  profile_matrix,
  complexes,
  method = c("pearson", "euclidean"),
  min_pairs = 10,
  bootstraps = 100,
  progress = TRUE
)
```

Arguments

profile_matrix	a matrix of chromatograms, with proteins in the rows and fractions in the columns, or a MSnSet object
complexes	a named list of protein complexes, where the name is the complex name and the entries are proteins within that complex
method	method to use to calculate edge weights; one of pearson or euclidean
min_pairs	the minimum number of pairwise observations to count a correlation or distance towards the z score
bootstraps	number of bootstraps to execute to estimate z scores
progress	whether to show the progress of the function

Value

a named vector of z scores for each complex in the input list

Examples

```
data(scott)
data(gold_standard)
complexes <- gold_standard[lengths(gold_standard) >= 3]
z_scores <- detect_complexes(t(scott), complexes)
length(na.omit(z_scores)) ## number of complexes that could be tested
z_scores[which.max(z_scores)] ## most significant complex
```

---

filter_profiles *Filter a co-elution profile matrix*

Description

Filter a matrix of co-elution/co-fractionation profiles by removing profiles without a certain number of non-mising or consecutive points.

Usage

```
filter_profiles(profile_matrix, min_points = 1, min_consecutive = 5)
```

Arguments

profile_matrix	a numeric matrix of co-elution profiles, with proteins in rows, or a MSnSet object
min_points	filter profiles without at least this many total, non-missing points
min_consecutive	filter profiles without at least this many consecutive, non-missing points

Value

the filtered profile matrix

Examples

```
data(scott)
nrow(scott)
filtered <- filter_profiles(scott)
nrow(scott)
```

fit_curve *Output the fit curve for a given mixture of Gaussians*

Description

For a Gaussian mixture model fit to a curve by `fit_gaussians`, output the fit curve using the coefficients rather than the `nls` object. This allows individual Gaussians to be removed from the fit model: for example, if their height is below a certain threshold, or their centres are outside the bounds of the chromatogram.

Usage

```
fit_curve(coef, indices)
```

Arguments

`coef`	numeric vector of coefficients for a Gaussian mixture model fit by `fit_gaussians`. This function assumes that the heights of the Gaussians are specified by coefficients beginning with "A" ("A1", "A2", "A3", etc.), centres are specified by coefficients beginning with "mu", and standard deviations are specified by coefficients beginning with "sigma".
`indices`	the indices, or x-values, to predict a fitted curve for (for example, the fractions in a given chromatogram)

Value

the fitted curve

Examples

```
data(scott)
chrom <- clean_profile(scott[1, ])
fit <- fit_gaussians(chrom, n_gaussians = 1)
curve <- fit_curve(fit$coefs, seq_along(chrom))
```

fit_gaussians *Fit a mixture of Gaussians to a chromatogram curve*

Description

Fit mixtures of one or more Gaussians to the curve formed by a chromatogram profile, using non-linear least-squares.

Usage

```
fit_gaussians(
  chromatogram,
  n_gaussians,
  max_iterations = 10,
  min_R_squared = 0.5,
  method = c("guess", "random"),
  filter_gaussians_center = TRUE,
  filter_gaussians_height = 0.15,
  filter_gaussians_variance_min = 0.1,
  filter_gaussians_variance_max = 50
)
```

Arguments

`chromatogram`	a numeric vector corresponding to the chromatogram trace
`n_gaussians`	the number of Gaussians to fit
`max_iterations`	the number of times to try fitting the curve with different initial conditions; defaults to 10
`min_R_squared`	the minimum R-squared value to accept when fitting the curve with different initial conditions; defaults to 0.5
`method`	the method used to select the initial conditions for nonlinear least squares optimization (one of "guess" or "random"); see `make_initial_conditions` for details
`filter_gaussians_center`	true or false: filter Gaussians whose centres fall outside the bounds of the chromatogram
`filter_gaussians_height`	Gaussians whose heights are below this fraction of the chromatogram height will be filtered. Setting this value to zero disables height-based filtering of fit Gaussians
`filter_gaussians_variance_min`	Gaussians whose variance falls below this number of fractions will be filtered. Setting this value to zero disables filtering.
`filter_gaussians_variance_max`	Gaussians whose variance is above this number of fractions will be filtered. Setting this value to zero disables filtering.

Value

a list with six entries: the number of Gaussians used to fit the curve; the R^2 of the fit; the number of iterations used to fit the curve with different initial conditions; the coefficients of the fit model; and the fit curve predicted by the fit model.

Examples

```
data(scott)
chrom <- clean_profile(scott[1, ])
fit <- fit_gaussians(chrom, n_gaussians = 1)
```

`gold_standard` *Reference set of human protein complexes*

Description

A reference set of 467 experimentally confirmed human protein complexes, derived from the EBI Complex Portal database.

Usage

```
data(gold_standard)
```

Format

a list containing 467 entries (character vectors)

Details

467 protein complexes, ranging in size from 2 to 44 proteins and involving 877 proteins in total, to provide a reference set of true positive and true negative interactions (intra- and inter-complex interactions, respectively) for demonstration in PrInCE analysis of a co-elution dataset. Other "gold standards" are possible in practice, most notably the CORUM database; however, the Complex Portal reference set is included in this package due to its CC-BY license.

Source

https://www.ebi.ac.uk/complexportal/complex/organisms

`impute_neighbors` *Impute single missing values*

Description

Impute single missing values within a chromatogram profile as the average of their neighbors.

Usage

```
impute_neighbors(chromatogram)
```

Arguments

chromatogram	a numeric vector corresponding to the chromatogram trace

Value

the imputed chromatogram

Examples

```
data(scott)
chrom <- scott[16, ]
imputed <- impute_neighbors(chrom)
```

---

is_unweighted *Test whether a network is unweighted*

Description

Test whether a network is unweighted

Usage

```
is_unweighted(network)
```

Arguments

network	the network to analyze

Value

true if the input network is a square logical or numeric matrix

Examples

```
data(gold_standard)
adj <- adjacency_matrix_from_list(gold_standard)
is_unweighted(adj) ## returns TRUE
```

is_weighted *Test whether a network is weighted*

Description

Test whether a network is weighted

Usage

```
is_weighted(network)
```

Arguments

network	the network to analyze

Value

true if the input network is a square numeric matrix with more than two values

Examples

```
data(gold_standard)
adj <- adjacency_matrix_from_list(gold_standard)
is_weighted(adj) ## returns FALSE
```

kristensen *Interactome of HeLa cells*

Description

Co-elution profiles derived from size exclusion chromatography (SEC) of HeLa cell lysates.

Usage

```
data(kristensen)
```

Format

a data frame with 1875 rows and 48 columns, with proteins in rows and SEC fractions in columns

Details

Protein quantitation was accomplished by SILAC (stable isotopic labelling by amino acids in cell culture), and is ratiometric, i.e., it reflects the ratio between the intensity of the heavy isotope and the light isotope ("H/L"). The dataset was initially described in Kristensen et al., *Nat. Methods* 2012. The medium isotope channel from replicate 1 (Supplementary Table 1a in the online supplementary information) is included in the `PrInCE` package. The R script used to generate this matrix from the supplementary materials of the paper is provided in the `data-raw` directory of the package source code.

Source

https://www.nature.com/articles/nmeth.2131

`kristensen_gaussians` *Fitted Gaussian mixture models for the* `kristensen` *dataset*

Description

The `kristensen` dataset consists of protein co-migration profiles derived from size exclusion chromatography (SEC) of unstimulated HeLa cell lysates. The `kristensen_gaussians` object contains Gaussian mixture models fit by the function `build_gaussians`; this is bundled with the R package in order to expedite the demonstration code, as the process of Gaussian fitting is one of the more time-consuming aspects of the package.

Usage

```
data(kristensen_gaussians)
```

Format

a named list with 1117 entries; names are proteins, and list items conain information about fitted Gaussians in the format that PrInCE expects

Details

As with the `kristensen` dataset, the code used to generate this data object is provided in the `data-raw` directory of the package source.

`make_feature_from_data_frame` *Create a feature vector for a classifier from a data frame*

Description

Convert a data frame containing pairwise interactions, and a score or other data associated with each interaction, into a feature vector that matches the dimensions of a data frame used as input to a classifier, such as a naive Bayes, random forests, or support vector machine classifier.

Usage

```
make_feature_from_data_frame(
  dat,
  target,
  dat_node_cols = c(1, 2),
  target_node_cols = c(1, 2),
  feature_col = 3,
  default_value = NA
)
```

Arguments

`dat`	a data frame containing pairwise interactions and a feature to be converted to a vector in a third column
`target`	the data frame of features that will be provided as input to a classifier
`dat_node_cols`	a vector of length two, denoting either the indices (integer vector) or column names (character vector) of the columns within the feature data frame; defaults to the first two columns of the data frame (`c(1, 2)`)
`target_node_cols`	a vector of length two, denoting either the indices (integer vector) or column names (character vector) of the columns within the target data frame; defaults to the first two columns of the data frame (`c(1, 2)`)
`feature_col`	the name or index of the column in the first data frame that contains a feature for each interaction
`default_value`	the default value for protein pairs that are not in the first data frame (set, by default, to `NA`)

Value

a vector matching the dimensions and order of the feature data frame, to use as input for a classifier in interaction prediction

`make_feature_from_expression`

Create a feature vector from expression data

Description

Convert a gene or protein expression matrix into a feature vector that matches the dimensions of a data frame used as input to a classifier, such as a naive Bayes, random forests, or support vector machine classifier, by calculating the correlation between each pair of genes or proteins.

Usage

```
make_feature_from_expression(expr, dat, node_columns = c(1, 2), ...)
```

Arguments

`expr`	a matrix containing gene or protein expression data, with genes/proteins in columns and samples in rows
`dat`	the data frame of features to be used by the classifier, with protein pairs in the columns specified by the `node_columns` argument
`node_columns`	a vector of length two, denoting either the indices (integer vector) or column names (character vector) of the columns within the data frame containing the nodes participating in pairwise interactions; defaults to the first two columns of the data frame (`c(1, 2)`)
`...`	arguments passed to `cor`

Value

a vector matching the dimensions and order of the feature data frame, to use as input for a classifier in interaction prediction

---

`make_initial_conditions`

Make initial conditions for curve fitting with a mixture of Gaussians

Description

Construct a set of initial conditions for curve fitting using nonlinear least squares using a mixture of Gaussians. The "guess" method ports code from the Matlab release of PrInCE. This method finds local maxima within the chromatogram, orders them by their separation (in number of fractions) from the previous local maxima, and uses the positions and heights of these local maxima (+/- some random noise) as initial conditions for Gaussian curve-fitting. The "random" method simply picks random values within the fraction and intensity intervals as starting points for Gaussian curve-fitting. The initial value of sigma is set by default to a random number within +/- 0.5 of two for both modes; this is based on our manual inspection of a large number of chromatograms.

Usage

```
make_initial_conditions(
  chromatogram,
  n_gaussians,
  method = c("guess", "random"),
  sigma_default = 2,
  sigma_noise = 0.5,
  mu_noise = 1.5,
  A_noise = 0.5
)
```

Arguments

chromatogram	a numeric vector corresponding to the chromatogram trace
n_gaussians	the number of Gaussians being fit
method	one of "guess" or "random", discussed above
sigma_default	the default mean initial value of sigma
sigma_noise	the amount of random noise to add or subtract from the default mean initial value of sigma
mu_noise	the amount of random noise to add or subtract from the Gaussian centers in "guess" mode
A_noise	the amount of random noise to add or subtract from the Gaussian heights in "guess" mode

Value

a list of three numeric vectors (A, mu, and sigma), each having a length equal to the maximum number of Gaussians to fit

Examples

```
data(scott)
chrom <- clean_profile(scott[16, ])
set.seed(0)
start <- make_initial_conditions(chrom, n_gaussians = 2, method = "guess")
```

---

make_labels *Make labels for a classifier based on a gold standard*

Description

Create labels for a classifier for protein pairs in the same order as in a dataset that will be used as input to a classifier, in a memory-friendly way.

Usage

```
make_labels(gold_standard, dat, node_columns = c(1, 2), protein_groups = NULL)
```

Arguments

gold_standard	an adjacency matrix of gold-standard interactions
dat	a data frame with interacting proteins in the first two columns
node_columns	a vector of length two, denoting either the indices (integer vector) or column names (character vector) of the columns within the data frame containing the nodes participating in pairwise interactions; defaults to the first two columns of the data frame (c(1, 2))
protein_groups	optionally, specify a list linking each protein in the first two columns of the input data frame to a protein group

Value

a vector of the same length as the input dataset, containing NAs for protein pairs not in the gold standard and ones or zeroes based on the content of the adjacency matrix

Examples

```
data(gold_standard)
adj <- adjacency_matrix_from_list(gold_standard)
proteins <- unique(unlist(gold_standard))
dat <- data.frame(protein_A = sample(proteins, 10),
                  protein_B = sample(proteins, 10))
labels <- make_labels(adj, dat)
```

---

match_matrix_dimensions

Match the dimensions of a query matrix to a profile matrix

Description

Match the row and column names of a square feature matrix to the row names of a profile matrix, adding rows/columns containing NAs when proteins in the profile matrix are missing from the feature matrix.

Usage

```
match_matrix_dimensions(query, profile_matrix)
```

Arguments

query	a square matrix containing features for pairs of proteins
profile_matrix	the profile matrix for which interactions are being predicted

Value

a square matrix with the same row and column names as the input profile matrix, for use in interaction prediction

Examples

```
data(gold_standard)
subset <- adjacency_matrix_from_list(gold_standard[seq(1, 200)])
target <- adjacency_matrix_from_list(gold_standard)
matched <- match_matrix_dimensions(subset, target)
dim(subset)
dim(target)
dim(matched)
```

---

predict_ensemble *Predict interactions using an ensemble of classifiers*

Description

Use an ensemble of classifiers to predict interactions from co-elution dataset features. The ensemble approach ensures that results are robust to the partitioning of the dataset into folds. For each model, the median of classifier scores across all folds is calculated. Then, the median of all such medians across all models is calculated.

Usage

```
predict_ensemble(
  dat,
  labels,
  classifier = c("NB", "SVM", "RF", "LR"),
  models = 1,
  cv_folds = 10,
  trees = 500,
  node_columns = c(1, 2)
)
```

Arguments

dat	a data frame containing interacting gene/protein pairs in the first two columns, and the features to use for classification in the remaining columns
labels	labels for each interaction in dat: 0 for negatives, 1 for positives, and NA for interactions outside the reference set
classifier	the type of classifier to use; one of "NB" (naive Bayes), "SVM" (support vector machine), "RF" (random forest), or "LR" (logistic regression)
models	the number of classifiers to train

cv_folds	the number of folds to split the reference dataset into when training each classifier. By default, each classifier uses ten-fold cross-validation, i.e., the classifier is trained on 90% of the dataset and used to classify the remaining 10%
trees	for random forest classifiers only, the number of trees to grow for each fold
node_columns	a vector of length two, denoting either the indices (integer vector) or column names (character vector) of the columns within the input data frame containing the nodes participating in pairwise interactions; defaults to the first two columns of the data frame (`c(1, 2)`)

Value

the input data frame of pairwise interactions, ranked by the median of classifier scores across all ensembled models

Examples

```
## calculate features
data(scott)
data(scott_gaussians)
subset <- scott[seq_len(500), ] ## limit to first 500 proteins
gauss <- scott_gaussians[names(scott_gaussians) %in% rownames(subset)]
features <- calculate_features(subset, gauss)
## make training labels
data(gold_standard)
ref <- adjacency_matrix_from_list(gold_standard)
labels <- make_labels(ref, features)
## predict interactions with naive Bayes classifier
ppi <- predict_ensemble(features, labels, classifier = "NB",
                        cv_folds = 3, models = 1)
```

---

predict_interactions *Predict interactions given a set of features and examples*

Description

Discriminate interacting from non-interacting protein pairs by training a machine learning model on a set of labelled examples, given a set of features derived from a co-elution profile matrix (see calculate_features.

Usage

```
predict_interactions(
  features,
  gold_standard,
  classifier = c("NB", "SVM", "RF", "LR", "ensemble"),
  verbose = FALSE,
  models = 10,
```

```
  cv_folds = 10,
  trees = 500
)
```

Arguments

features	a data frame with proteins in the first two columns, and features to be passed to the classifier in the remaining columns
gold_standard	an adjacency matrix of "gold standard" interactions used to train the classifier
classifier	the type of classifier to use: one of "NB" (naive Bayes), "SVM" (support vector machine), "RF" (random forest), "LR" (logistic regression), or "ensemble" (an ensemble of all four)
verbose	if TRUE, print a series of messages about the stage of the analysis
models	the number of classifiers to train and average across, each with a different k-fold cross-validation split
cv_folds	the number of folds to use for k-fold cross-validation
trees	for random forests only, the number of trees in the forest

Details

PrInCE implements four different classifiers (naive Bayes, support vector machine, random forest, and logistic regression). Naive Bayes is used as a default. The classifiers are trained on the gold standards using a ten-fold cross-validation procedure, training on 90 that are part of the training data, the held-out split is used to assign a classifier score, whereas for the remaining protein pairs, the median of all ten folds is used. Furthermore, to ensure the results are not sensitive to the precise classifier split used, an ensemble of multiple classifiers (ten, by default) is trained, and the classifier score is subsequently averaged across classifiers.

PrInCE can also ensemble across multiple different types of classifiers, by supplying the "ensemble" option to the classifier argument.

Value

a ranked data frame of pairwise interactions, with the classifier score, label, and cumulative precision for each interaction

Examples

```
## calculate features
data(scott)
data(scott_gaussians)
subset <- scott[seq_len(500), ] ## limit to first 500 proteins
gauss <- scott_gaussians[names(scott_gaussians) %in% rownames(subset)]
features <- calculate_features(subset, gauss)
## load training data
data(gold_standard)
ref <- adjacency_matrix_from_list(gold_standard)
## predict interactions
ppi <- predict_interactions(features, ref, cv_folds = 3, models = 1)
```

PrInCE *PrInCE: Prediction of Interactomes from Co-Elution*

Description

PrInCE is a computational approach to infer protein-protein interaction networks from co-elution proteomics data, also called co-migration, co-fractionation, or protein correlation profiling. This family of methods separates interacting protein complexes on the basis of their diameter or biochemical properties. Protein-protein interactions can then be inferred for pairs of proteins with similar elution profiles. PrInCE implements a machine-learning approach to identify protein-protein interactions given a set of labelled examples, using features derived exclusively from the data. This allows PrInCE to infer high-quality protein interaction networks from raw proteomics data, without bias towards known interactions or functionally associated proteins, making PrInCE a unique resource for discovery.

Usage

```
PrInCE(
  profiles,
  gold_standard,
  gaussians = NULL,
  precision = NULL,
  verbose = FALSE,
  min_points = 1,
  min_consecutive = 5,
  min_pairs = 3,
  impute_NA = TRUE,
  smooth = TRUE,
  smooth_width = 4,
  max_gaussians = 5,
  max_iterations = 50,
  min_R_squared = 0.5,
  method = c("guess", "random"),
  criterion = c("AICc", "AIC", "BIC"),
  pearson_R_raw = TRUE,
  pearson_R_cleaned = TRUE,
  pearson_P = TRUE,
  euclidean_distance = TRUE,
  co_peak = TRUE,
  co_apex = TRUE,
  n_pairs = FALSE,
  classifier = c("NB", "SVM", "RF", "LR", "ensemble"),
  models = 1,
  cv_folds = 10,
  trees = 500
)
```

Arguments

profiles	the co-elution profile matrix, or a list of profile matrices if replicate experiments were performed. Can be a single numeric matrix, with proteins in rows and fractions in columns, or a list of matrices. Alternatively, can be provided as a single MSnSet object or a list of objects.
gold_standard	a set of 'gold standard' interactions, used to train the classifier. Can be provided either as an adjacency matrix, in which both rows and columns correspond to protein IDs in the co-elution matrix or matrices, or as a list of proteins in the same complex, which will be converted to an adjacency matrix by PrInCE. Zeroes in the adjacency matrix are interpreted by PrInCE as "true negatives" when calculating precision.
gaussians	optionally, provide Gaussian mixture models fit by the build_gaussians function. If profiles is a numeric matrix, this should be the named list output by build_gaussians for that matrix; if profiles is a list of numeric matrices, this should be a list of named lists
precision	optionally, return only interactions above the given precision; by default, all interactions are returned and the user can subsequently threshold the list using the threshold_precision function
verbose	if TRUE, print a series of messages about the stage of the analysis
min_points	filter profiles without at least this many total, non-missing points; passed to filter_profiles
min_consecutive	filter profiles without at least this many consecutive, non-missing points; passed to filter_profiles
min_pairs	minimum number of overlapping fractions between any given protein pair to consider a potential interaction
impute_NA	if true, impute single missing values with the average of neighboring values; passed to clean_profiles
smooth	if true, smooth the chromatogram with a moving average filter; passed to clean_profiles
smooth_width	width of the moving average filter, in fractions; passed to clean_profiles
max_gaussians	the maximum number of Gaussians to fit; defaults to 5. Note that Gaussian mixtures with more parameters than observed (i.e., non-zero or NA) points will not be fit. Passed to choose_gaussians
max_iterations	the number of times to try fitting the curve with different initial conditions; defaults to 50. Passed to fit_gaussians
min_R_squared	the minimum R-squared value to accept when fitting the curve with different initial conditions; defaults to 0.5. Passed to fit_gaussians
method	the method used to select the initial conditions for nonlinear least squares optimization (one of "guess" or "random"); see make_initial_conditions for details. Passed to fit_gaussians
criterion	the criterion to use for model selection; one of "AICc" (corrected AIC, and default), "AIC", or "BIC". Passed to choose_gaussians
pearson_R_raw	if true, include the Pearson correlation (R) between raw profiles as a feature

`pearson_R_cleaned`
: if true, include the Pearson correlation (R) between cleaned profiles as a feature

`pearson_P`
: if true, include the P-value of the Pearson correlation between raw profiles as a feature

`euclidean_distance`
: if true, include the Euclidean distance between cleaned profiles as a feature

`co_peak`
: if true, include the 'co-peak score' (that is, the distance, in fractions, between the single highest value of each profile) as a feature

`co_apex`
: if true, include the 'co-apex score' (that is, the minimum Euclidean distance between any pair of fit Gaussians) as a feature

`n_pairs`
: if `TRUE`, include the number of fractions in which both of a given pair of proteins were detected as a feature

`classifier`
: the type of classifier to use: one of `"NB"` (naive Bayes), `"SVM"` (support vector machine), `"RF"` (random forest), `"LR"` (logistic regression), or `"ensemble"` (an ensemble of all four)

`models`
: the number of classifiers to train and average across, each with a different k-fold cross-validation split

`cv_folds`
: the number of folds to use for k-fold cross-validation

`trees`
: for random forests only, the number of trees in the forest

Details

PrInCE takes as input a co-elution matrix, with detected proteins in rows and fractions as columns, and a set of 'gold standard' true positives and true negatives. If replicate experiments were performed, a list of co-elution matrices can be provided as input. PrInCE will construct features for each replicate separately and use features from all replicates as input to the classifier. The 'gold standard' can be either a data frame or adjacency matrix of known interactions (and non-interactions), or a list of protein complexes. For computational convenience, Gaussian mixture models can be pre-fit to every profile and provided separately to the `PrInCE` function. The matrix, or matrices, can be provided to PrInCE either as numeric matrices or as `MSnSet` objects.

PrInCE implements three different types of classifiers to predict protein-protein interaction networks, including naive Bayes (the default), random forests, and support vector machines. The classifiers are trained on the gold standards using a ten-fold cross-validation procedure, training on 90 that are part of the training data, the held-out split is used to assign a classifier score, whereas for the remaining protein pairs, the median of all ten folds is used. Furthermore, to ensure the results are not sensitive to the precise classifier split used, an ensemble of multiple classifiers (ten, by default) is trained, and the classifier score is subsequently averaged across classifiers. PrInCE can also ensemble across a set of classifiers.

By default, PrInCE calculates six features from each pair of co-elution profiles as input to the classifier, including conventional similarity metrics but also several features specifically adapted to co-elution proteomics. For example, one such feature is derived from fitting a Gaussian mixture model to each elution profile, then calculating the smallest Euclidean distance between any pair of fitted Gaussians. The complete set of features includes:

1. the Pearson correlation between raw co-elution profiles;
2. the p-value of the Pearson correlation between raw co-elution profiles;

3. the Pearson correlation between cleaned profiles, which are generated by imputing single missing values with the mean of their neighbors, replacing remaining missing values with random near-zero noise, and smoothing the profiles using a moving average filter (see `clean_profile`);
4. the Euclidean distance between cleaned profiles;
5. the 'co-peak' score, defined as the distance, in fractions, between the maximum values of each profile; and
6. the 'co-apex' score, defined as the minimum Euclidean distance between any pair of fit Gaussians

The output of PrInCE is a ranked data frame, containing the classifier score for every possible protein pair. PrInCE also calculates the precision at every point in this ranked list, using the 'gold standard' set of protein complexes or binary interactions. Our recommendation is to select a threshold for the precision and use this to construct an unweighted protein interaction network.

Value

a ranked data frame of interacting proteins, with the precision at each point in the list

References

Stacey RG, Skinnider MA, Scott NE, Foster LJ (2017). "A rapid and accurate approach for prediction of interactomes from co-elution data (PrInCE)." *BMC Bioinformatics*, **18**(1), 457.

Scott NE, Brown LM, Kristensen AR, Foster LJ (2015). "Development of a computational framework for the analysis of protein correlation profiling and spatial proteomics experiments." *Journal of Proteomics*, **118**, 112–129.

Kristensen AR, Gsponer J, Foster LJ (2012). "A high-throughput approach for measuring temporal changes in the interactome." *Nature Methods*, **9**(9), 907–909.

Skinnider MA, Stacey RG, Foster LJ (2018). "Genomic data integration systematically biases interactome mapping." *PLoS Computational Biology*, **14**(10), e1006474.

Examples

```
data(scott)
data(scott_gaussians)
data(gold_standard)
# analyze only the first 100 profiles
subset <- scott[seq_len(500), ]
gauss <- scott_gaussians[names(scott_gaussians) %in% rownames(subset)]
ppi <- PrInCE(subset, gold_standard,
  gaussians = gauss, models = 1,
  cv_folds = 3
)
```

replace_missing_data *Replace missing data with median ± random noise*

Description

Replace missing data within each numeric column of a data frame with the column median, plus or minus some random noise, in order to train classifiers that do not easily ignore missing data (e.g. random forests or support vector machines).

Usage

```
replace_missing_data(dat, noise_pct = 0.05)
```

Arguments

dat	the data frame to replace missing data in
noise_pct	the standard deviation of the random normal distribution from which to draw added noise, expressed as a percentage of the standard deviation of the non-missing values in each column

Value

a data frame with missing values in each numeric column replaced by the column median, plus or minus some random noise

scott *Cytoplasmic interactome of Jurkat T cells during apoptosis*

Description

Co-elution profiles derived from size exclusion chromatography (SEC) of cytoplasmic fractions from Jurkat T cells, 4 hours following Fas stimulation.

Usage

```
data(scott)
```

Format

a data frame with 1560 rows and 55 columns, with proteins in rows and SEC fractions in columns

Details

Protein quantitation was accomplished by SILAC (stable isotopic labelling by amino acids in cell culture), and is ratiometric, i.e., it reflects the ratio between the intensity of the heavy isotope and the light isotope ("H/L"). The dataset was initially described in Scott et al., *Mol. Syst. Biol.* 2017. The heavy isotope channel from replicate 1 is included in the `PrInCE` package. The R script used to generate this matrix from the supplementary materials of the paper is provided in the `data-raw` directory of the package source code.

Source

http://msb.embopress.org/content/13/1/906

---

scott_gaussians *Fitted Gaussian mixture models for the* scott *dataset*

Description

The `scott` dataset consists of protein co-migration profiles derived from size exclusion chromatography (SEC) of cytoplasmic fractions from Jurkat T cells, 4 hours following Fas stimulation. The `scott_gaussians` object contains Gaussian mixture models fit by the function `build_gaussians`; this is bundled with the R package in order to expedite the demonstration code, as the process of Gaussian fitting is one of the more time-consuming aspects of the package.

Usage

```
data(scott_gaussians)
```

Format

a named list with 970 entries; names are proteins, and list items conain information about fitted Gaussians in the format that PrInCE expects

Details

As with the `scott` dataset, the code used to generate this data object is provided in the `data-raw` directory of the package source.

`threshold_precision` *Threshold interactions at a given precision cutoff*

Description

Threshold interactions at a given precision cutoff

Usage

```
threshold_precision(interactions, threshold)
```

Arguments

`interactions`	the ranked list of interactions output by `predict_interactions`, including a `precision` column
`threshold`	the minimum precision of the unweighted interaction network to return

Value

the subset of the original ranked list at the given precision

Examples

```
data(scott)
data(scott_gaussians)
data(gold_standard)
# analyze only the first 100 profiles
subset <- scott[seq_len(500), ]
gauss <- scott_gaussians[names(scott_gaussians) %in% rownames(subset)]
ppi <- PrInCE(subset, gold_standard,
  gaussians = gauss, models = 1,
  cv_folds = 3
)
network <- threshold_precision(ppi, threshold = 0.5)
nrow(network)
```

Index