Package: spqrp (via r-universe)

October 20, 2026

Title Sample Provenance Quality Resolver in Proteomics

Version 0.1.0

Description Detect sample-provenance inconsistencies and potential mix-ups in mass-spectrometry-based plasma-proteome cohorts. Provides a clustering-based approach (build a nearest-neighbour graph in a dimensionality-reduced space and iteratively split large components by edge weight), a threshold-based approach (classify sample pairs as belonging or not-belonging from a pairwise distance cutoff), parameter optimization over distance metrics and cutoffs, and a pairwise random-forest classifier for protein importance ranking. This is a native R port of the author's Python package 'spqrp' (<https://github.com/fhradilak/spqrp>), implementing methods from an associated manuscript currently in preparation.

License GPL-3

URL https://github.com/fhradilak/spqrp_r

BugReports https://github.com/fhradilak/spqrp_r/issues

Encoding UTF-8

LazyData true

Depends R (>= 4.5.0)

Imports cli, dplyr (>= 1.1.0), ggplot2 (>= 3.5.0), igraph (>= 2.0.0), lgr, pROC, randomForest, ranger (>= 0.16.0), rlang, solitude (>= 1.1.3), stats, tibble, tidyr (>= 1.3.0), utils, withr

Suggests knitr, plotly (>= 4.10.0), recipes, rmarkdown, smacof, testthat (>= 3.0.0), themis, uwot (>= 0.2.0), vdiffr

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libglpk-dev libicu-dev libxml2-dev

Repository https://fhradilak.r-universe.dev

Date/Publication 2026-06-22 09:27:55 UTC

RemoteUrl https://github.com/fhradilak/spqrp_r

RemoteRef HEAD

RemoteSha 724422e65321410329db3cf3f7620562ae683cd8

Contents

by_isolation_forest 2
check_input_data_format 4
cohort_a_ranking 4
filter_by_occurrence 5
log_transform 6
normalize_medianintensity 6
optimize_parameters 7
perform_distance_evaluation_on_ranked_proteins 8
plate_correct_residuals_by_protein 10
print.spqrp_train 11
remove_outlier_samples 12
retrieve_ranking 13
run_clustering 14
spqrp_example_data 15
spqrp_example_path 16
train_pairwise_balanced_rand_forest 17
train_with_normalise 18

Index **21**

`by_isolation_forest` *Isolation Forest outlier detection*

Description

Pivots to a (sample x protein) matrix, runs an Isolation Forest via the ‘solitude‘ package (a pure-R port of the same Liu et al. 2008 algorithm that scikit-learn’s ‘IsolationForest‘ uses), and returns the data frame with outlier rows removed plus a tibble of per-sample anomaly scores.

Usage

```
by_isolation_forest(
  protein_df,
  peptide_df = NULL,
  n_estimators = 100L,
  impute_zero = FALSE,
  impute_median = FALSE,
  outlier_threshold = 0.6,
  contamination = "auto",
```

```
  quiet = TRUE
)
```

Arguments

protein_df	Long-format intensity data frame.
peptide_df	Optional peptide-level data frame; subset alongside ‘protein_df‘ using the same outlier list.
n_estimators	Number of trees.
impute_zero	Replace NA intensities with 0 before fitting.
impute_median	Replace NA intensities with column-wise median.
outlier_threshold	Used when ‘contamination = "auto"‘. Anomaly score above which a sample is flagged. Default ‘0.6‘ (calibrated for solitude’s score scale; on sklearn’s scale this would be ‘0.5‘).
contamination	Either ‘"auto"‘ (default; use ‘outlier_threshold‘) or a numeric in ‘[0, 1]‘ specifying the fraction of the data to flag as outliers (top-by-score). Mirrors sklearn’s ‘IsolationForest(contamination=...)‘ API.
quiet	If ‘TRUE‘ (default), suppress informational status messages. Set ‘FALSE‘ to print progress and per-call summaries (sample counts, chosen cutoff, etc.). Warnings about genuine data issues – e.g. samples dropped from the analysis – are emitted regardless.

Details

Two ways to decide which samples are outliers, mirroring sklearn’s ‘IsolationForest‘ API:

* ‘contamination = "auto"‘ (default) – flag every sample whose anomaly score exceeds ‘outlier_threshold‘.
* ‘contamination‘ set to a numeric in ‘[0, 1]‘ – flag exactly the top ‘contamination * 100‘ ‘outlier_threshold‘. Mirrors sklearn’s ‘IsolationForest(contamination = 0.1)‘.

On the **sklearn score scale**, ‘contamination = "auto"‘ corresponds to a threshold of ‘0.5‘. solitude’s scores, however, are systematically shifted upward because ‘solitude‘ (via ‘ranger‘) uses ‘mtry = ncol - 1‘ and ‘extratrees‘ split bounds drawn from the full dataset rather than from the per-tree subsample. The result is that inlier scores typically sit between ‘0.55‘ and ‘0.60‘ even on clean data, so the sklearn-calibrated ‘0.5‘ cutoff would flag everything. The default ‘outlier_threshold = 0.6‘ below is calibrated empirically for solitude’s distribution and reproduces sklearn’s "few-to-zero outliers on clean data" behaviour. Lower it (e.g. ‘0.55‘) for more aggressive flagging, or use ‘contamination‘ for a percentile-based rule.

Value

Invisibly returns a named list with ‘protein_df‘, ‘peptide_df‘, ‘outlier_list‘, ‘anomaly_df‘, and possibly ‘messages‘ on failure. ‘invisible()‘ keeps the REPL silent on unassigned calls; assign the result to a name and inspect with ‘result$protein_df‘ etc.

Examples

```
df <- spqrp_example_data("input_cohort_df")
res <- by_isolation_forest(df, impute_median = TRUE)
res$outlier_list
```

`check_input_data_format` *Validate required columns of a cohort and (optionally) a ranking*

Description

Throws an informative error when required columns are missing.

Usage

```
check_input_data_format(df, importance_ranking = NULL)
```

Arguments

`df`	Cohort data frame; must contain 'Sample_ID', 'Patient_ID', 'Protein', 'Intensity'.
`importance_ranking`	Optional ranking data frame; must contain 'Protein', 'Importance' if supplied.

Value

Invisible 'TRUE'.

Examples

```
df <- spqrp_example_data("input_cohort_df")
ranking <- spqrp_example_data("protein_ranking")
check_input_data_format(df, ranking)
```

`cohort_a_ranking` *Protein-importance ranking for plasma cohort "A"*

Description

A pre-computed protein-importance ranking produced by the pairwise balanced random-forest classifier ([train_pairwise_balanced_rand_forest()]) on a real mass-spectrometry plasma-proteome cohort. It serves as the built-in default ranking for [perform_distance_evaluation_on_ranked_proteins()] and [optimize_parameters()] when the caller supplies neither 'top_importance_df' nor 'top_importance_path'.

Usage

```
cohort_a_ranking
```

Format

A [tibble][tibble::tibble] with one row per protein and two columns:

Protein Character. Protein identifier (UniProt accession with gene suffix, e.g. ‘"P01861_IGHG4"‘).

Importance Numeric. Random-forest importance score; higher means more discriminative. Rows are ordered from most to least important.

Source

Pairwise balanced random-forest importances computed on plasma cohort "A", derived from mass-spectrometry plasma-proteome measurements.

See Also

[perform_distance_evaluation_on_ranked_proteins()], [optimize_parameters()], [retrieve_ranking()]

Examples

```
head(cohort_a_ranking)
```

---

filter_by_occurrence *Keep proteins present in at least a given fraction of samples*

Description

Keep proteins present in at least a given fraction of samples

Usage

```
filter_by_occurrence(df, cutoff = 0.7)
```

Arguments

df	Long-format intensity data frame.
cutoff	Fraction in ‘[0, 1]‘. A protein is kept when its non-NA intensity covers at least ‘cutoff‘ of the samples in ‘df‘.

Value

Filtered tibble in the same shape as ‘df‘.

Examples

```
df <- spqrp_example_data("input_cohort_df")
kept <- filter_by_occurrence(df, cutoff = 0.7)
length(unique(kept$Protein))
```

log_transform *Log2-transform the intensity column in place (long format)*

Description

Log2-transform the intensity column in place (long format)

Usage

```
log_transform(df)
```

Arguments

df	Long-format intensity data frame.

Value

'df' with 'Intensity = log2(Intensity)'.

Examples

```
df <- spqrp_example_data("input_cohort_df")
head(log_transform(df))
```

normalize_medianintensity *Per-sample median normalisation (log-space subtraction)*

Description

Subtracts each sample's median intensity from its intensities, then re-centers on the dataset's overall median. By default 'df' is assumed to already be in log space. If 'revert_log = TRUE', the function reverts the log transform first, then divides by the per-sample median ratio.

Usage

```
normalize_medianintensity(
  dataset,
  string_of_pool = "",
  revert_log = FALSE,
  sample = SAMPLE,
  plot = TRUE
)
```

Arguments

dataset	Long-format intensity data frame.
string_of_pool	If non-empty, samples whose ID contains this substring are excluded from normalization (kept out of the post-normalization data).
revert_log	If ‘TRUE‘, run [revert_log_transform()] first.
sample	Column to group by (defaults to ‘"Sample_ID"‘).
plot	If ‘TRUE‘, attach a before/after boxplot.

Details

Returns a list with ‘data‘ (the normalized tibble) and ‘plot‘ (a ggplot showing before/after boxplots).

Value

List with ‘data‘ and (optionally) ‘plot‘.

Examples

```
df <- spqrp_example_data("input_cohort_df")
norm <- normalize_medianintensity(log_transform(df), plot = FALSE)
head(norm$data)
```

---

optimize_parameters *Grid-search the cutoff that optimises a chosen performance metric*

Description

For each value of ‘n‘ (the number of top-ranked proteins) and each fractional-p (only used when ‘metric = "fractional"‘), sweeps a fixed grid of percentile cutoffs and records the parameters that optimize ‘optimization_strategy‘.

Usage

```
optimize_parameters(
  df,
  metric = "correlation",
  log_file = NULL,
  top_importance_path = NULL,
  top_importance_df = NULL,
  range = 2:49,
  optimization_strategy = default_strategies(),
  remove_list = character(),
  quiet = TRUE
)
```

Arguments

df	Long-format cohort data frame.
metric	Distance metric. ‘"fractional"‘ enables a sweep over ‘fractional_p_values‘.
log_file	Optional path; if non-NULL the optimization log is written there. Default ‘NULL‘ (no log).
top_importance_path	Optional CSV path with ‘Protein‘, ‘Importance‘. Used only when ‘top_importance_df‘ is ‘NULL‘.
top_importance_df	Optional pre-loaded ranking. When both this and ‘top_importance_path‘ are ‘NULL‘ (the default) the bundled [cohort_a_ranking] dataset is used.
range	Integer vector of ‘n‘ values to evaluate.
optimization_strategy	One of ‘"fp+fn"‘, ‘"fp"‘, ‘"fn"‘, ‘"F1"‘, ‘"precision"‘, ‘"sensitivity"‘. Optimizes for the lowest false negative (fn) or false positive (fp) scores or for the highest F1, precision, sensitivity.
remove_list	Proteins to drop from the ranking.
quiet	If ‘TRUE‘ (default), suppress informational status messages. Set ‘FALSE‘ to print progress and per-call summaries (sample counts, chosen cutoff, etc.). Warnings about genuine data issues – e.g. samples dropped from the analysis – are emitted regardless.

Value

Tibble of one row per ‘n‘, listing the best parameters and their classification metrics.

Examples

```
df      <- spqrp_example_data("input_cohort_df")
ranking <- spqrp_example_data("protein_ranking")
best <- optimize_parameters(
  df = df, top_importance_df = ranking,
  metric = "manhattan", range = 2:4
)
best
```

---

perform_distance_evaluation_on_ranked_proteins

Threshold-based pairwise distance evaluation

Description

Computes pairwise distances on the top-‘n‘ proteins, splits sample pairs by a percentile cutoff (‘p‘) on the distance distribution, and computes classification metrics against the patient ID ground truth.

Usage

```
perform_distance_evaluation_on_ranked_proteins(
  df,
  top_importance_path = NULL,
  top_importance_df = NULL,
  n = 10L,
  p = 0.5,
  remove_list = NULL,
  metric = "correlation",
  fractional_p = 0.5,
  threshold_based = TRUE,
  quiet = TRUE,
  number_display_neighbours = 4L,
  name = "",
  plot = TRUE,
  save_path = NULL
)
```

Arguments

df	Long-format cohort data frame.
top_importance_path	Optional path to a CSV with ‘Protein‘ and ‘Importance‘. Used only when ‘top_importance_df‘ is ‘NULL‘.
top_importance_df	Optional pre-loaded ranking data frame. If supplied, ‘top_importance_path‘ is ignored. When both are ‘NULL‘ (the default) the bundled [cohort_a_ranking] dataset is used.
n	Number of top-ranked proteins.
p	Percentile (0-100) for the distance cutoff.
remove_list	Proteins to exclude from the ranking.
metric	Distance metric (see [get_distances()]).
fractional_p	Fractional/Minkowski exponent.
threshold_based	If ‘FALSE‘, only return distances and skip classification.
quiet	If ‘TRUE‘ (default), suppress informational status messages. Set ‘FALSE‘ to print progress and per-call summaries (sample counts, chosen cutoff, etc.). Warnings about genuine data issues – e.g. samples dropped from the analysis – are emitted regardless.
number_display_neighbours	Number of nearest neighbours to report.
name	Plot title suffix; appended to "Distribution of Pairwise Distances". Set this to a cohort label (e.g. ‘name = "Cohort A"‘) so saved plots are self-documenting.
plot	If ‘TRUE‘, draw the distance histogram with FN/FP overlays and a legend matching the Python figure.

save_path — Where to save a high-resolution render of the distance-distribution plot. Accepts ‘NULL‘ (default, don’t save), ‘TRUE‘ (auto-save to a timestamped file in ‘tempdir()‘), or a character path (e.g. ‘"distances.png"‘). Same semantics as [run_clustering()]’s ‘save_path‘. Only used when ‘plot = TRUE‘.

Value

Invisibly returns a list with ‘top_importance‘, ‘nearest_neighbours‘, ‘cutoff‘, ‘belonging‘, ‘not_belonging‘, ‘eval_metrics‘, ‘distance_matrix‘, and ‘plot‘ (the ggplot built when ‘plot = TRUE‘; ‘NULL‘ otherwise). ‘invisible()‘ keeps the REPL silent on unassigned calls. Assign to a name and use ‘result$plot‘, ‘result$eval_metrics‘, etc. To render the distance-distribution histogram on demand: ‘print(result$plot)‘.

Examples

```
df      <- spqrp_example_data("input_cohort_df")
ranking <- spqrp_example_data("protein_ranking")
result <- perform_distance_evaluation_on_ranked_proteins(
  df = df, top_importance_df = ranking,
  metric = "manhattan", p = 0.989, n = 4L
)
result$eval_metrics[c("TP", "FP", "FN", "TN", "F1")]
result$plot
```

---

plate_correct_residuals_by_protein

Regress intensity on plate and replace it with OLS residuals

Description

Identifies a ‘plate‘ (or ‘Plate‘) column, encodes it as integers, fits ‘lm(Intensity ~ plate)‘, and replaces ‘Intensity‘ with the model’s residuals. If no plate column is present, returns the input unchanged with a message.

Usage

```
plate_correct_residuals_by_protein(
  group_data,
  individual = PATIENT,
  sample = SAMPLE,
  impute = FALSE,
  verbose = FALSE
)
```

Arguments

group_data	Long-format intensity data frame.
individual	Patient identifier column (default ‘"Patient_ID"‘).
sample	Sample identifier column (default ‘"Sample_ID"‘).
impute	If ‘TRUE‘, impute missing intensities by patient/protein median before regression; otherwise drop NA rows.
verbose	If ‘TRUE‘, also build before/after boxplots.

Value

Tibble with corrected ‘Intensity‘. Attribute ‘"plot"‘ carries the diagnostic ggplot when ‘verbose = TRUE‘.

Examples

```
df <- spqrp_example_data("input_cohort_df")
df$plate <- rep(c("A", "B"), length.out = nrow(df))
corrected <- plate_correct_residuals_by_protein(df)
head(corrected)
```

---

print.spqrp_train *Print a one-line summary of an spqrp_train object*

Description

Displays the classifier backend, the number of training/test pairs, and the feature count for the pairwise random-forest model returned by [train_with_normalise()] and [train_pairwise_balanced_rand_forest()].

Usage

```
## S3 method for class 'spqrp_train'
print(x, ...)
```

Arguments

x	A ‘spqrp_train‘ object.
...	Unused; present for S3 generic compatibility.

Value

‘x‘, invisibly.

`remove_outlier_samples` *Remove samples flagged as outliers by Isolation Forest*

Description

Convenience wrapper around [by_isolation_forest()] with median imputation. Removes samples (not proteins) whose intensity profile looks anomalous compared to the rest of the cohort.

Usage

```
remove_outlier_samples(
  df,
  sample = SAMPLE,
  contamination = "auto",
  outlier_threshold = 0.6,
  quiet = TRUE
)
```

Arguments

`df`	Long-format intensity data frame.
`sample`	Sample column (defaults to ‘"Sample_ID"‘).
`contamination`	‘"auto"‘ (default) or a numeric in ‘[0, 1]‘. See [by_isolation_forest()] for details.
`outlier_threshold`	Anomaly-score cutoff used when ‘contamination = "auto"‘. Default ‘0.6‘, calibrated empirically for solitude’s anomaly-score distribution. See [by_isolation_forest()] for the rationale.
`quiet`	If ‘TRUE‘ (default), suppress informational status messages. Set ‘FALSE‘ to print progress and per-call summaries (sample counts, chosen cutoff, etc.). Warnings about genuine data issues – e.g. samples dropped from the analysis – are emitted regardless.

Details

Pass ‘contamination = 0.1‘ (or any fraction) to mimic sklearn’s ‘IsolationForest(contamination = 0.1)‘ behaviour, or keep the default ‘contamination = "auto"‘ to use the conservative absolute threshold.

The returned list includes ‘anomaly_plot‘, a ‘plotly‘ bar chart of per-sample anomaly scores coloured by outlier flag. Printing the object at the R REPL (or ‘print(result$anomaly_plot)‘ inside a script) renders the chart – mirroring the Python wrapper’s auto-shown bar plot, but without surprising side effects when the function is called non-interactively.

Value

Invisibly returns a named list with components: * ‘df‘ – filtered tibble (same shape as ‘df‘, fewer rows) * ‘anomaly_df‘ – per-sample tibble of ‘Sample_ID‘, ‘Anomaly Score‘, ‘Outlier‘ * ‘outlier_list‘ – character vector of flagged ‘Sample_ID‘s * ‘anomaly_plot‘ – a ‘plotly‘ figure; ‘print(result$anomaly_plot)‘ to view the bar chart. ‘NULL‘ if the optional ‘plotly‘ package is not installed (a message explains how to enable it).

The return is wrapped in ‘invisible()‘ so unassigned REPL calls stay silent (matches ‘quiet = TRUE‘). Assign to a name to inspect.

Examples

```
df <- spqrp_example_data("input_cohort_df")
filtered <- remove_outlier_samples(df, contamination = "auto")
filtered$outlier_list
head(filtered$df)
```

`retrieve_ranking` *Convert classifier output to a Protein / Importance ranking*

Description

Strips the ‘diff_‘ prefix the pairwise model adds to feature names. Importance values are normalised to sum to 1.0 across features at training time (matching sklearn’s ‘clf.feature_importances_‘ convention), so the numbers in the returned tibble are directly comparable to Python output. Rank order is preserved across the normalisation.

Usage

```
retrieve_ranking(results)
```

Arguments

`results`	Output of [train_with_normalise()].

Value

Tibble with ‘Protein‘ and ‘Importance‘ columns; ‘Importance‘ sums to ~1.0.

Examples

```
df <- spqrp_example_data("input_cohort_df")
results <- train_with_normalise(df, plate_corrected = FALSE,
                                outlier_removal = FALSE)
retrieve_ranking(results)
```

`run_clustering` *End-to-end clustering pipeline*

Description

Computes pairwise distances on the top-‘n‘ ranked proteins, builds a k-nearest-neighbour graph in a 2D embedding (default UMAP), iteratively splits big components by max-weight edge, and visualises the result.

Usage

```
run_clustering(
  df,
  ranking,
  n_neighbors,
  max_component_size,
  metric = "manhattan",
  n = 20L,
  fractional_p = 0.98,
  plot_name = "DF_Ranking_X on DF_Y",
  method = "UMAP",
  figsize = c(16, 16),
  dpi = 200L,
  save_path = NULL,
  quiet = TRUE
)
```

Arguments

`df`	Long-format cohort data frame.
`ranking`	Data frame with ‘Protein‘ and ‘Importance‘.
`n_neighbors`	Number of nearest-neighbour edges per sample.
`max_component_size`	Maximum allowed connected component size.
`metric`	Distance metric.
`n`	Number of top-ranked proteins to use.
`fractional_p`	Fractional/Minkowski exponent.
`plot_name`	Plot title.
`method`	Dimensionality reduction method (‘"UMAP"‘, ‘"PCA"‘, ‘"MDS"‘).
`figsize`	Numeric vector of length 2: width and height in inches. Used both for ‘ggsave‘ (when ‘save_path‘ is set) and to auto-scale point sizes, line widths, and text on the plot. Larger values produce more readable plots. Default ‘c(16, 16)‘.
`dpi`	Resolution (dots per inch) for the saved file. Default ‘200‘ (matches Python matplotlib’s default-ish output; bump to 300 for print).

save_path — Where to save a high-resolution PNG/SVG/PDF render. Accepts: * ‘NULL‘ (default) – don’t save; only return the ggplot object. The function still prints a hint about how to download the plot. * a character path (e.g. ‘"out.png"‘ or ‘"figs/cluster.svg"‘) – save there via ‘ggsave()‘. Extension chooses the format.

quiet — If ‘TRUE‘ (default), suppress informational status messages. Set ‘FALSE‘ to print progress and per-call summaries (sample counts, chosen cutoff, etc.). Warnings about genuine data issues – e.g. samples dropped from the analysis – are emitted regardless.

Value

Invisibly returns a list with ‘result_filtered‘, ‘G‘ (the igraph object), ‘cluster_assignments‘, ‘transitive_results‘, ‘uncertain_samples‘, ‘error_candidate_samples‘, ‘plot‘, and ‘saved_path‘ (the path passed in via ‘save_path‘, or ‘NULL‘). ‘invisible()‘ keeps the REPL silent on unassigned calls. Assign to a name to inspect; render the cluster plot on demand via ‘print(result$plot)‘.

Examples

```
df      <- spqrp_example_data("input_cohort_df")
ranking <- spqrp_example_data("protein_ranking")
res <- run_clustering(
  df = df, ranking = ranking,
  n_neighbors = 1L, max_component_size = 2L,
  metric = "manhattan", method = "PCA"
)
head(res$cluster_assignments)
res$transitive_results
```

---

spqrp_example_data *Load a bundled example data file as a tibble*

Description

Load a bundled example data file as a tibble

Usage

```
spqrp_example_data(which = c("input_cohort_df", "protein_ranking"))
```

Arguments

which — One of ‘"input_cohort_df"‘, ‘"protein_ranking"‘.

Details

The package ships two example CSV files in ‘inst/extdata/‘, both describing a small synthetic cohort intended only for runnable examples and tests:

* ‘example_input_cohort_df.csv‘ – mock cohort (30 patients x 2 samples x 5 proteins) in long format with the required columns ‘Sample_ID‘, ‘Patient_ID‘, ‘Protein‘, ‘Intensity‘. * ‘example_protein_ranking.csv‘ – protein importance ranking aligned with the mock cohort.

The real-cohort protein-importance ranking is provided separately as the lazy-loaded [cohort_a_ranking] dataset: a tibble of ‘Protein‘ / ‘Importance‘ computed by the pairwise balanced random-forest classifier on plasma cohort "A". It is the built-in default ranking for [perform_distance_evaluation_on_ranked_proteins()] and [optimize_parameters()], and is accessed with ‘data(cohort_a_ranking)‘ or ‘spqrp::cohort_a_ranking‘ rather than through this function.

Use [spqrp_example_path()] if you need the file path instead of the loaded data.

Value

A tibble.

Examples

```
spqrp_example_data("input_cohort_df")
```

`spqrp_example_path` *Filesystem path to a bundled example CSV*

Description

Filesystem path to a bundled example CSV

Usage

```
spqrp_example_path(which = c("input_cohort_df", "protein_ranking"))
```

Arguments

`which`	One of ‘"input_cohort_df"‘, ‘"protein_ranking"‘.

Value

Absolute character path inside ‘inst/extdata/‘.

Examples

```
spqrp_example_path("input_cohort_df")
```

`train_pairwise_balanced_rand_forest` *Pairwise balanced random-forest classifier*

Description

Builds a pairwise design matrix (feature-wise differences of every sample pair, optionally augmented with the Euclidean distance), labels each pair 1 if the two samples share a patient ID, then trains a class- balanced random forest. The classifier backend is selectable.

Usage

```
train_pairwise_balanced_rand_forest(
  X_train,
  y_train,
  X_test,
  y_test,
  df_pivot_test,
  compute_euclid = TRUE,
  method = "F1",
  classifier_backend = c("randomForest", "ranger", "themis_smote"),
  k = 0L,
  plots_per_sample = FALSE,
  sample_decision_curve = FALSE,
  absolute = FALSE,
  quiet = TRUE
)
```

Arguments

`X_train`, `X_test`	Sample x feature matrices.
`y_train`, `y_test`	Patient labels (vectors with one entry per row).
`df_pivot_test`	Wide test frame including ‘Sample_ID‘ column – used to label misclassified pairs by sample.
`compute_euclid`	Add a NaN-aware Euclidean distance feature.
`method`	Threshold selection (see [get_threshold()]).
`classifier_backend`	‘"randomForest"‘ (default – closest behaviour to Python’s ‘imblearn.BalancedRandomForestClassifier‘ via per-tree balanced bootstrap), ‘"ranger"‘ (faster; class-weighted impurity), or ‘"themis_smote"‘ (SMOTE oversampling). See <https://github.com/fhradilak/spqrp_r/blob/main/articles/divergence.md> for the tradeoffs. Importance values returned in the results are normalised to sum to 1.0 across features (matching sklearn’s ‘clf.feature_importances_‘ convention) regardless of backend.
`k`	Fold number for diagnostic printing.

plots_per_sample
: Per-sample probability plots.

sample_decision_curve
: If 'TRUE', draw ROC + PR + threshold plots.

absolute
: Take absolute value of feature differences before passing to the model. (Stored after training is complete.)

quiet
: If 'TRUE' (default), suppress informational status messages. Set 'FALSE' to print progress and per-call summaries (sample counts, chosen cutoff, etc.). Warnings about genuine data issues – e.g. samples dropped from the analysis – are emitted regardless.

Value

Named list as described in the package docs.

Examples

```
df <- spqrp_example_data("input_cohort_df")
# In practice, call the high-level [train_with_normalise()] instead --
# it handles the train/test split, normalisation, and pivoting for you.:
res <- train_with_normalise(df, plate_corrected = FALSE,
                            outlier_removal = FALSE)
res$classifier_backend
```

---

train_with_normalise *End-to-end ranking pipeline: filter, normalise, optionally plate-correct, train RF*

Description

Mirrors 'protein_selection.train_with_normalise' from the Python package but exposes 'classifier_backend' so users can compare three RF variants ('"ranger"', '"randomForest"', '"themis_smote"'). See <https://github.com/fhradilak/spqrp_r/blob/main/articles/numerical-divergence.md> for the trade-offs.

Usage

```
train_with_normalise(
  df,
  threshold = 0.7,
  test_size = 0.3,
  plate_corrected = TRUE,
  individual = PATIENT,
  sample = SAMPLE,
  compute_euclid = FALSE,
  method = "F1",
```

```
  outlier_removal = TRUE,
  train_individuals = NULL,
  test_individuals = NULL,
  sample_decision_curve = FALSE,
  classifier_backend = c("randomForest", "ranger", "themis_smote"),
  importance_method = "impurity",
  plot_per_sample = FALSE,
  absolute = FALSE,
  quiet = TRUE
)
```

Arguments

`df`	Long-format cohort data frame.
`threshold`	Occurrence-filter threshold.
`test_size`	Patient-level test fraction.
`plate_corrected`	If ‘TRUE‘, run plate-effect residualisation.
`individual`	Patient column.
`sample`	Sample column.
`compute_euclid`	Add NaN-aware Euclidean distance feature.
`method`	Threshold-selection strategy.
`outlier_removal`	Run [by_isolation_forest()] on each split.
`train_individuals`, `test_individuals`	Explicit split overrides.
`sample_decision_curve`	Draw ROC/PR curves.
`classifier_backend`	‘"randomForest"‘ (default – closest behaviour to Python’s ‘imblearn.BalancedRandomForestClassifier‘), ‘"ranger"‘ (faster), or ‘"themis_smote"‘. The default was changed from ‘"ranger"‘ to ‘"randomForest"‘ to bring R rankings closer to the Python port. See <https://github.com/fhradilak/spqrp divergence.md>.
`importance_method`	Unused placeholder (kept for API parity).
`plot_per_sample`	Per-sample probability plots.
`absolute`	Use absolute pairwise differences.
`quiet`	If ‘TRUE‘ (default), suppress informational status messages (train/test split listing, "Proteins only in test set", outliers removed, fold headers, per-fold metrics, top-importance list, and per-misclassified-pair prints) and skip auto-rendering of the ROC / PR / probability plots. Set ‘FALSE‘ to print everything. Warnings about genuine data issues are emitted regardless.

Value

'spqrp_train' S3 object (a named list with classifier, pair indices, feature importances, misclassified pairs).

Examples

```
df <- spqrp_example_data("input_cohort_df")
res <- train_with_normalise(df, plate_corrected = FALSE,
                            outlier_removal = FALSE)
retrieve_ranking(res)
```

Index

∗ **datasets**
cohort_a_ranking, 4

by_isolation_forest, 2

check_input_data_format, 4
cohort_a_ranking, 4

filter_by_occurrence, 5

log_transform, 6

normalize_medianintensity, 6

optimize_parameters, 7

perform_distance_evaluation_on_ranked_proteins, 8
plate_correct_residuals_by_protein, 10
print.spqrp_train, 11

remove_outlier_samples, 12
retrieve_ranking, 13
run_clustering, 14

spqrp_example_data, 15
spqrp_example_path, 16

train_pairwise_balanced_rand_forest, 17
train_with_normalise, 18