Package ‘metasnf’

March 25, 2025

Title Meta Clustering with Similarity Network Fusion

Version 2.0.6

Description Framework to facilitate patient subtyping with similarity network fusion and meta clustering. The similarity network fusion (SNF) algorithm was introduced by Wang et al. (2014) in <doi:10.1038/nmeth.2810>. SNF is a data integration approach that can transform high-dimensional and diverse data types into a single similarity network suitable for clustering with minimal loss of information from each initial data source. The meta clustering approach was introduced by Caruana et al. (2006) in <doi:10.1109/ICDM.2006.103>. Meta clustering involves generating a wide range of cluster solutions by adjusting clustering hyperparameters, then clustering the solutions themselves into a manageable number of qualitatively similar solutions, and finally characterizing representative solutions to find ones that are best for the user's specific context. This package provides a framework to easily transform multi-modal data into a wide range of similarity network fusion-derived cluster solutions as well as to visualize, characterize, and validate those solutions. Core package functionality includes easy customization of distance metrics, clustering algorithms, and SNF hyperparameters to generate diverse clustering solutions; calculation and plotting of associations between features, between patients, and between cluster solutions; and standard cluster validation approaches including resampled measures of cluster stability, standard metrics of cluster quality, and label propagation to evaluate generalizability in unseen data. Associated vignettes guide the user through using the package to identify patient subtypes while adhering to best practices for unsupervised learning.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Imports cli, cluster, data.table, digest, dplyr, ggplot2, grDevices, MASS, mclust, methods, progressr, purrr, RColorBrewer, rlang, SNFtool, stats, tibble, tidyr, utils

Suggests circlize, ComplexHeatmap, InteractiveComplexHeatmap, clv, future, future.apply, knitr, rmarkdown, testthat (>= 3.0.0), ggalluvial, lifecycle, dbscan

Config/testthat/edition 3

Depends R (>= 4.1.0)

LazyData true

VignetteBuilder knitr

URL https://branchlab.github.io/metasnf/,
https://github.com/BRANCHlab/metasnf/

BugReports https://github.com/BRANCHlab/metasnf/issues

NeedsCompilation no

Author Prashanth S Velayudhan [aut, cre],
Xiaoqiao Xu [aut],
Prajkta Kallurkar [aut],
Ana Patricia Balbon [aut],
Maria T Secara [aut],
Adam Taback [aut],
Denise Sabac [aut],
Nicholas Chan [aut],
Shihao Ma [aut],
Bo Wang [aut],
Daniel Felsky [aut],
Stephanie H Ameis [aut],
Brian Cox [aut],
Colin Hawco [aut],
Lauren Erdman [aut],
Anne L Wheeler [aut, ths]

Maintainer Prashanth S Velayudhan <psvelayu@gmail.com>

Repository CRAN

Date/Publication 2025-03-25 21:20:02 UTC

Contents

`abcd_anxiety` *Mock ABCD anxiety data*

Description

A randomly shuffled and anonymized copy of anxiety data from the NIMH Data archive. The original file used was pdem02.txt. The file was pre-processed by the abcdutils package (https://github.com/BRANCHlab/abcdutils)
function `get_cbcl_anxiety`.

Usage

```
abcd_anxiety
```

Format

`abcd_anxiety`**:**

A data frame with 275 rows and 2 columns:

patient The unique identifier of the ABCD dataset

cbcl_anxiety_r Ordinal value of impairment on CBCL anxiety, either 0 (no impairment), 1 (borderline clinical), or 2 (clinically impaired)

Source

Though this data is no longer "real" ABCD data, the reference for using ABCD as a data source is below:

Data used in the preparation of this article were obtained from the Adolescent Brain Cognitive DevelopmentSM (ABCD) Study (https://abcdstudy.org), held in the NIMH Data Archive (NDA). This is a multisite, longitudinal study designed to recruit more than 10,000 children age 9-10 and follow them over 10 years into early adulthood. The ABCD Study® is supported by the National Institutes of Health and additional federal partners under award numbers U01DA041048, U01DA050989, U01DA051016, U01DA041022, U01DA051018, U01DA051037, U01DA050987, U01DA041174, U01DA041106, U01DA041117, U01DA041028, U01DA041134, U01DA050988, U01DA051039, U01DA041156, U01DA041025, U01DA041120, U01DA051038, U01DA041148, U01DA041093, U01DA041089, U24DA041123, U24DA041147. A full list of supporters is available at https://abcdstudy.org/federal-partners.html. A listing of participating sites and a complete listing of the study investigators can be found at https://abcdstudy.org/consortium_members/. ABCD consortium investigators designed and implemented the study and/or provided data but did not necessarily participate in the analysis or writing of this report. This manuscript reflects the views of the authors and may not reflect the opinions or views of the NIH or ABCD consortium investigators.

`abcd_colour` *Mock ABCD "colour" data*

Description

A randomly shuffled and anonymized copy of depression data from the NIMH Data archive. The original file used was pdem02.txt. The file was pre-processed by the abcdutils package (https://github.com/BRANCHlab/abcd function `get_cbcl_depress`. The data was transformed into categorical colour values to demonstrate the Chi-squared test capabilities of `extend_solutions`.

Usage

```
abcd_colour
```

Format

`abcd_colour`**:**

A data frame with 275 rows and 2 columns:

patient The unique identifier of the ABCD dataset

colour Categorical transformation of `cbcl_depress`.

Source

Though this data is no longer "real" ABCD data, the reference for using ABCD as a data source is below:

Data used in the preparation of this article were obtained from the Adolescent Brain Cognitive DevelopmentSM (ABCD) Study (https://abcdstudy.org), held in the NIMH Data Archive (NDA). This is a multisite, longitudinal study designed to recruit more than 10,000 children age 9-10 and follow them over 10 years into early adulthood. The ABCD Study® is supported by the National Institutes of Health and additional federal partners under award numbers U01DA041048, U01DA050989, U01DA051016, U01DA041022, U01DA051018, U01DA051037, U01DA050987, U01DA041174, U01DA041106, U01DA041117, U01DA041028, U01DA041134, U01DA050988, U01DA051039, U01DA041156, U01DA041025, U01DA041120, U01DA051038, U01DA041148, U01DA041093, U01DA041089, U24DA041123, U24DA041147. A full list of supporters is available at https://abcdstudy.org/federal-partners.html. A listing of participating sites and a complete listing of the study investigators can be found at https://abcdstudy.org/consortium_members/. ABCD consortium investigators designed and implemented the study and/or provided data but did not necessarily participate in the analysis or writing of this report. This manuscript reflects the views of the authors and may not reflect the opinions or views of the NIH or ABCD consortium investigators.

`abcd_cort_sa` *Mock ABCD cortical surface area data*

Description

A randomly shuffled and anonymized copy of cortical surface area data from the NIMH Data archive. The original file used was mrisdp10201.txt The file was pre-processed by the abcdutils package (https://github.com/BRANCHlab/abcdutils) function `get_cort_t`.

Usage

```
abcd_cort_sa
```

Format

`abcd_cort_sa`**:**

A data frame with 188 rows and 152 columns:

patient The unique identifier of the ABCD dataset

... Cortical surface areas of various ROIs (mm^2, I think)

Source

Though this data is no longer "real" ABCD data, the reference for using ABCD as a data source is below:

Data used in the preparation of this article were obtained from the Adolescent Brain Cognitive DevelopmentSM (ABCD) Study (https://abcdstudy.org), held in the NIMH Data Archive (NDA). This is a multisite, longitudinal study designed to recruit more than 10,000 children age 9-10 and follow them over 10 years into early adulthood. The ABCD Study® is supported by the National Institutes of Health and additional federal partners under award numbers U01DA041048, U01DA050989, U01DA051016, U01DA041022, U01DA051018, U01DA051037, U01DA050987, U01DA041174, U01DA041106, U01DA041117, U01DA041028, U01DA041134, U01DA050988, U01DA051039, U01DA041156, U01DA041025, U01DA041120, U01DA051038, U01DA041148, U01DA041093, U01DA041089, U24DA041123, U24DA041147. A full list of supporters is available at https://abcdstudy.org/federal-partners.html. A listing of participating sites and a complete listing of the study investigators can be found at https://abcdstudy.org/consortium_members/. ABCD consortium investigators designed and implemented the study and/or provided data but did not necessarily participate in the analysis or writing of this report. This manuscript reflects the views of the authors and may not reflect the opinions or views of the NIH or ABCD consortium investigators.

abcd_cort_t *Mock ABCD cortical thickness data*

Description

A randomly shuffled and anonymized copy of cortical thickness data from the NIMH Data archive. The original file used was mrisdp10201.txt The file was pre-processed by the abcdutils package (https://github.com/BRANCHlab/abcdutils) function `get_cort_t`.

Usage

```
abcd_cort_t
```

Format

`abcd_cort_t`:

A data frame with 188 rows and 152 columns:

patient The unique identifier of the ABCD dataset

... Cortical thicknesses of various ROIs (mm^3, I think)

Source

Though this data is no longer "real" ABCD data, the reference for using ABCD as a data source is below:

Data used in the preparation of this article were obtained from the Adolescent Brain Cognitive DevelopmentSM (ABCD) Study (https://abcdstudy.org), held in the NIMH Data Archive (NDA). This is a multisite, longitudinal study designed to recruit more than 10,000 children age 9-10 and follow them over 10 years into early adulthood. The ABCD Study® is supported by the National Institutes of Health and additional federal partners under award numbers U01DA041048, U01DA050989, U01DA051016, U01DA041022, U01DA051018, U01DA051037, U01DA050987, U01DA041174, U01DA041106, U01DA041117, U01DA041028, U01DA041134, U01DA050988, U01DA051039, U01DA041156, U01DA041025, U01DA041120, U01DA051038, U01DA041148, U01DA041093, U01DA041089, U24DA041123, U24DA041147. A full list of supporters is available at https://abcdstudy.org/federal-partners.html. A listing of participating sites and a complete listing of the study investigators can be found at https://abcdstudy.org/consortium_members/. ABCD consortium investigators designed and implemented the study and/or provided data but did not necessarily participate in the analysis or writing of this report. This manuscript reflects the views of the authors and may not reflect the opinions or views of the NIH or ABCD consortium investigators.

abcd_depress *Mock ABCD depression data*

Description

A randomly shuffled and anonymized copy of depression data from the NIMH Data archive. The original file used was pdem02.txt. The file was pre-processed by the abcdutils package (https://github.com/BRANCHlab/abcd function `get_cbcl_depress`.

Usage

```
abcd_depress
```

Format

`abcd_depress`**:**

A data frame with 275 rows and 2 columns:

patient The unique identifier of the ABCD dataset

cbcl_depress_r Ordinal value of impairment on CBCL anxiety, either 0 (no impairment), 1 (borderline clinical), or 2 (clinically impaired)

Source

Though this data is no longer "real" ABCD data, the reference for using ABCD as a data source is below:

Data used in the preparation of this article were obtained from the Adolescent Brain Cognitive DevelopmentSM (ABCD) Study (https://abcdstudy.org), held in the NIMH Data Archive (NDA). This is a multisite, longitudinal study designed to recruit more than 10,000 children age 9-10 and follow them over 10 years into early adulthood. The ABCD Study® is supported by the National Institutes of Health and additional federal partners under award numbers U01DA041048, U01DA050989, U01DA051016, U01DA041022, U01DA051018, U01DA051037, U01DA050987, U01DA041174, U01DA041106, U01DA041117, U01DA041028, U01DA041134, U01DA050988, U01DA051039, U01DA041156, U01DA041025, U01DA041120, U01DA051038, U01DA041148, U01DA041093, U01DA041089, U24DA041123, U24DA041147. A full list of supporters is available at https://abcdstudy.org/federal-partners.html. A listing of participating sites and a complete listing of the study investigators can be found at https://abcdstudy.org/consortium_members/. ABCD consortium investigators designed and implemented the study and/or provided data but did not necessarily participate in the analysis or writing of this report. This manuscript reflects the views of the authors and may not reflect the opinions or views of the NIH or ABCD consortium investigators.

`abcd_h_income` *Mock ABCD income data*

Description

Like abcd_income, but with no NAs in patient column

Usage

```
abcd_h_income
```

Format

`abcd_income`**:**

A data frame with 300 rows and 2 columns:

patient The unique identifier of the ABCD dataset

household_income Household income in 3 category levels (low = 1, medium = 2, high = 3)

Source

Though this data is no longer "real" ABCD data, the reference for using ABCD as a data source is below:

Data used in the preparation of this article were obtained from the Adolescent Brain Cognitive DevelopmentSM (ABCD) Study (https://abcdstudy.org), held in the NIMH Data Archive (NDA). This is a multisite, longitudinal study designed to recruit more than 10,000 children age 9-10 and follow them over 10 years into early adulthood. The ABCD Study® is supported by the National Institutes of Health and additional federal partners under award numbers U01DA041048, U01DA050989, U01DA051016, U01DA041022, U01DA051018, U01DA051037, U01DA050987, U01DA041174, U01DA041106, U01DA041117, U01DA041028, U01DA041134, U01DA050988, U01DA051039, U01DA041156, U01DA041025, U01DA041120, U01DA051038, U01DA041148, U01DA041093, U01DA041089, U24DA041123, U24DA041147. A full list of supporters is available at https://abcdstudy.org/federal-partners.html. A listing of participating sites and a complete listing of the study investigators can be found at https://abcdstudy.org/consortium_members/. ABCD consortium investigators designed and implemented the study and/or provided data but did not necessarily participate in the analysis or writing of this report. This manuscript reflects the views of the authors and may not reflect the opinions or views of the NIH or ABCD consortium investigators.

`abcd_income` *Mock ABCD income data*

Description

A randomly shuffled and anonymized copy of income data from the NIMH Data archive. The original file used was pdem02.txt The file was pre-processed by the abcdutils package (https://github.com/BRANCHlab/abcdutils) function `get_income`.

Usage

```
abcd_income
```

Format

`abcd_income`**:**

A data frame with 300 rows and 2 columns:

patient The unique identifier of the ABCD dataset

household_income Household income in 3 category levels (low = 1, medium = 2, high = 3)

Source

Though this data is no longer "real" ABCD data, the reference for using ABCD as a data source is below:

Data used in the preparation of this article were obtained from the Adolescent Brain Cognitive DevelopmentSM (ABCD) Study (https://abcdstudy.org), held in the NIMH Data Archive (NDA). This is a multisite, longitudinal study designed to recruit more than 10,000 children age 9-10 and follow them over 10 years into early adulthood. The ABCD Study® is supported by the National Institutes of Health and additional federal partners under award numbers U01DA041048, U01DA050989, U01DA051016, U01DA041022, U01DA051018, U01DA051037, U01DA050987, U01DA041174, U01DA041106, U01DA041117, U01DA041028, U01DA041134, U01DA050988, U01DA051039, U01DA041156, U01DA041025, U01DA041120, U01DA051038, U01DA041148, U01DA041093, U01DA041089, U24DA041123, U24DA041147. A full list of supporters is available at https://abcdstudy.org/federal-partners.html. A listing of participating sites and a complete listing of the study investigators can be found at https://abcdstudy.org/consortium_members/. ABCD consortium investigators designed and implemented the study and/or provided data but did not necessarily participate in the analysis or writing of this report. This manuscript reflects the views of the authors and may not reflect the opinions or views of the NIH or ABCD consortium investigators.

`abcd_pubertal` *Mock ABCD pubertal status data*

Description

A randomly shuffled and anonymized copy of pubertal status data from the NIMH Data archive. The original files used were abcd_ssphp01.txt and abcd_ssphy01.txt. The file was pre-processed by the abcdutils package (https://github.com/BRANCHlab/abcdutils) function `get_pubertal_status`.

Usage

```
abcd_pubertal
```

Format

`abcd_pubertal`**:**

A data frame with 275 rows and 2 columns:

patient The unique identifier of the ABCD dataset

pubertal_status Average reported pubertal status between child and parent (1-5 categorical scale)

Source

Though this data is no longer "real" ABCD data, the reference for using ABCD as a data source is below:

Data used in the preparation of this article were obtained from the Adolescent Brain Cognitive DevelopmentSM (ABCD) Study (https://abcdstudy.org), held in the NIMH Data Archive (NDA). This is a multisite, longitudinal study designed to recruit more than 10,000 children age 9-10 and follow them over 10 years into early adulthood. The ABCD Study® is supported by the National Institutes of Health and additional federal partners under award numbers U01DA041048, U01DA050989, U01DA051016, U01DA041022, U01DA051018, U01DA051037, U01DA050987, U01DA041174, U01DA041106, U01DA041117, U01DA041028, U01DA041134, U01DA050988, U01DA051039, U01DA041156, U01DA041025, U01DA041120, U01DA051038, U01DA041148, U01DA041093, U01DA041089, U24DA041123, U24DA041147. A full list of supporters is available at https://abcdstudy.org/federal-partners.html. A listing of participating sites and a complete listing of the study investigators can be found at https://abcdstudy.org/consortium_members/. ABCD consortium investigators designed and implemented the study and/or provided data but did not necessarily participate in the analysis or writing of this report. This manuscript reflects the views of the authors and may not reflect the opinions or views of the NIH or ABCD consortium investigators.

`abcd_subc_v` *Mock ABCD subcortical volumes data*

Description

A randomly shuffled and anonymized copy of subcortical volume data from the NIMH Data archive. The original file used was smrip10201.txt The file was pre-processed by the abcdutils package (https://github.com/BRANCHlab/abcdutils) function `get_subc_v`.

Usage

```
abcd_subc_v
```

Format

`abcd_subc_v`**:**

A data frame with 174 rows and 31 columns:

patient The unique identifier of the ABCD dataset

... Subcortical volumes of various ROIs (mm^3, I think)

Source

Though this data is no longer "real" ABCD data, the reference for using ABCD as a data source is below:

Data used in the preparation of this article were obtained from the Adolescent Brain Cognitive DevelopmentSM (ABCD) Study (https://abcdstudy.org), held in the NIMH Data Archive (NDA). This is a multisite, longitudinal study designed to recruit more than 10,000 children age 9-10 and

follow them over 10 years into early adulthood. The ABCD Study® is supported by the National Institutes of Health and additional federal partners under award numbers U01DA041048, U01DA050989, U01DA051016, U01DA041022, U01DA051018, U01DA051037, U01DA050987, U01DA041174, U01DA041106, U01DA041117, U01DA041028, U01DA041134, U01DA050988, U01DA051039, U01DA041156, U01DA041025, U01DA041120, U01DA051038, U01DA041148, U01DA041093, U01DA041089, U24DA041123, U24DA041147. A full list of supporters is available at https://abcdstudy.org/federal-partners.html. A listing of participating sites and a complete listing of the study investigators can be found at https://abcdstudy.org/consortium_members/. ABCD consortium investigators designed and implemented the study and/or provided data but did not necessarily participate in the analysis or writing of this report. This manuscript reflects the views of the authors and may not reflect the opinions or views of the NIH or ABCD consortium investigators.

`add_settings_df_rows` *Add rows to a settings_df*

Description

Add rows to a settings_df

Usage

```
add_settings_df_rows(
  sdf,
  n_solutions = 0,
  min_removed_inputs = 0,
  max_removed_inputs = sum(startsWith(colnames(sdf), "inc_")) - 1,
  dropout_dist = "exponential",
  min_alpha = NULL,
  max_alpha = NULL,
  min_k = NULL,
  max_k = NULL,
  min_t = NULL,
  max_t = NULL,
  alpha_values = NULL,
  k_values = NULL,
  t_values = NULL,
  possible_snf_schemes = c(1, 2, 3),
  clustering_algorithms = NULL,
  continuous_distances = NULL,
  discrete_distances = NULL,
  ordinal_distances = NULL,
  categorical_distances = NULL,
  mixed_distances = NULL,
  dfl = NULL,
  snf_input_weights = NULL,
  snf_domain_weights = NULL,
  retry_limit = 10,
  allow_duplicates = FALSE
)
```

Arguments

`sdf` The existing settings data frame

`n_solutions` Number of rows to generate for the settings data frame.

`min_removed_inputs` The smallest number of input data frames that may be randomly removed. By default, 0.

`max_removed_inputs` The largest number of input data frames that may be randomly removed. By default, this is 1 less than all the provided input data frames in the data list.

`dropout_dist` Parameter controlling how the random removal of input data frames should occur. Can be "none" (no input data frames are randomly removed), "uniform" (uniformly sample between min_removed_inputs and max_removed_inputs to determine number of input data frames to remove), or "exponential" (pick number of input data frames to remove by sampling from min_removed_inputs to max_removed_inputs with an exponential distribution; the default).

`min_alpha` The minimum value that the alpha hyperparameter can have. Random assigned value of alpha for each row will be obtained by uniformly sampling numbers between `min_alpha` and `max_alpha` at intervals of 0.1. Cannot be used in conjunction with the `alpha_values` parameter.

`max_alpha` The maximum value that the alpha hyperparameter can have. See `min_alpha` parameter. Cannot be used in conjunction with the `alpha_values` parameter.

`min_k` The minimum value that the k hyperparameter can have. Random assigned value of k for each row will be obtained by uniformly sampling numbers between `min_k` and `max_k` at intervals of 1. Cannot be used in conjunction with the `k_values` parameter.

`max_k` The maximum value that the k hyperparameter can have. See `min_k` parameter. Cannot be used in conjunction with the `k_values` parameter.

`min_t` The minimum value that the t hyperparameter can have. Random assigned value of t for each row will be obtained by uniformly sampling numbers between `min_t` and `max_t` at intervals of 1. Cannot be used in conjunction with the `t_values` parameter.

`max_t` The maximum value that the t hyperparameter can have. See `min_t` parameter. Cannot be used in conjunction with the `t_values` parameter.

`alpha_values` A number or numeric vector of a set of possible values that alpha can take on. Value will be obtained by uniformly sampling the vector. Cannot be used in conjunction with the `min_alpha` or `max_alpha` parameters.

`k_values` A number or numeric vector of a set of possible values that k can take on. Value will be obtained by uniformly sampling the vector. Cannot be used in conjunction with the `min_k` or `max_k` parameters.

`t_values` A number or numeric vector of a set of possible values that t can take on. Value will be obtained by uniformly sampling the vector. Cannot be used in conjunction with the `min_t` or `max_t` parameters.

`possible_snf_schemes`
: A vector containing the possible snf_schemes to uniformly randomly select from. By default, the vector contains all 3 possible schemes: c(1, 2, 3). 1 corresponds to the "individual" scheme, 2 corresponds to the "domain" scheme, and 3 corresponds to the "twostep" scheme.

`clustering_algorithms`
: A list of clustering algorithms to uniformly randomly pick from when clustering. When not specified, randomly select between spectral clustering using the eigen-gap heuristic and spectral clustering using the rotation cost heuristic. See ?clust_fns_list for more details on running custom clustering algorithms.

`continuous_distances`
: A vector of continuous distance metrics to use when a custom dist_fns_list is provided.

`discrete_distances`
: A vector of categorical distance metrics to use when a custom dist_fns_list is provided.

`ordinal_distances`
: A vector of categorical distance metrics to use when a custom dist_fns_list is provided.

`categorical_distances`
: A vector of categorical distance metrics to use when a custom dist_fns_list is provided.

`mixed_distances`
: A vector of mixed distance metrics to use when a custom dist_fns_list is provided.

`dfl`
: List containing distance metrics to vary over. See ?generate_dist_fns_list.

`snf_input_weights`
: Nested list containing weights for when SNF is used to merge individual input measures (see ?generate_snf_weights)

`snf_domain_weights`
: Nested list containing weights for when SNF is used to merge domains (see ?generate_snf_weights)

`retry_limit`
: The maximum number of attempts to generate a novel row. This function does not return matrices with identical rows. As the range of requested possible settings tightens and the number of requested rows increases, the risk of randomly generating a row that already exists increases. If a new random row has matched an existing row `retry_limit` number of times, the function will terminate.

`allow_duplicates`
: If TRUE, enables creation of a settings data frame with duplicate non-feature weighting related hyperparameters. This function should only be used when paired with a custom weights matrix that has non-duplicate rows.

Value

A settings data frame

adjusted_rand_index_heatmap

Heatmap of pairwise adjusted rand indices between solutions

Description

[Deprecated] Defunct function to create an ARI heatmap. Please use `meta_cluster_heatmap()` instead.

Usage

```
adjusted_rand_index_heatmap(
  aris,
  order = NULL,
  cluster_rows = FALSE,
  cluster_columns = FALSE,
  log_graph = FALSE,
  scale_diag = "none",
  min_colour = "#282828",
  max_colour = "firebrick2",
  col = circlize::colorRamp2(c(min(aris), max(aris)), c(min_colour, max_colour)),
  ...
)
```

Arguments

`aris` Matrix of adjusted rand indices from `calc_aris()`

`order` Numeric vector containing row order of the heatmap.

`cluster_rows` Whether rows should be clustered.

`cluster_columns` Whether columns should be clustered.

`log_graph` If TRUE, log transforms the graph.

`scale_diag` Method of rescaling matrix diagonals. Can be "none" (don't change diagonals), "mean" (replace diagonals with average value of off-diagonals), or "zero" (replace diagonals with 0).

`min_colour` Colour used for the lowest value in the heatmap.

`max_colour` Colour used for the highest value in the heatmap.

`col` Colour ramp to use for the heatmap.

`...` Additional parameters passed to `similarity_matrix_heatmap()`, the function that this function wraps.

Value

Returns a heatmap (class "Heatmap" from package ComplexHeatmap) that displays the pairwise adjusted Rand indices (similarities) between the cluster solutions of the provided solutions data frame.

age_df *Mock age data*

Description

Mock age data

Usage

```
age_df
```

Format

`age_df`**:**

A data frame with 200 rows and 2 columns:

patient_id Random three-digit number uniquely identifying the patient

age Mock age feature

Source

This data came from the SNFtool package, with slight modifications.

alluvial_cluster_plot *Alluvial plot of patients across cluster counts and important features*

Description

This alluvial plot shows how observations in a similarity matrix could have been clustered over a set of clustering functions.

Usage

```
alluvial_cluster_plot(
  cluster_sequence,
  similarity_matrix,
  dl = NULL,
  data = NULL,
  key_outcome,
  key_label = key_outcome,
  extra_outcomes = NULL,
  title = NULL
)
```

Arguments

`cluster_sequence`
A list of clustering algorithms.

`similarity_matrix`
A similarity matrix.

`dl` A data list.

`data` A data frame that contains any features to include in the plot.

`key_outcome` The name of the feature that determines how each patient stream is coloured in the alluvial plot.

`key_label` Name of key outcome to be used for the plot legend.

`extra_outcomes` Names of additional features to add to the plot.

`title` Title of the plot.

Value

An alluvial plot (class "gg" and "ggplot") showing distribution of a feature across varying number cluster solutions.

Examples

```
input_dl <- data_list(
    list(gender_df, "gender", "demographics", "categorical"),
    list(diagnosis_df, "diagnosis", "clinical", "categorical"),
    uid = "patient_id"
)

sc <- snf_config(input_dl, n_solutions = 1)

sol_df <- batch_snf(input_dl, sc, return_sim_mats = TRUE)

sim_mats <- sim_mats_list(sol_df)

clust_fn_sequence <- list(spectral_two, spectral_four)

alluvial_cluster_plot(
    cluster_sequence = clust_fn_sequence,
    similarity_matrix = sim_mats[[1]],
    dl = input_dl,
    key_outcome = "gender", # the name of the feature of interest
    key_label = "Gender", # how the feature of interest should be displayed
    extra_outcomes = "diagnosis", # more features to plot but not colour by
    title = "Gender Across Cluster Counts"
)
```

`anxiety` *Mock ABCD anxiety data*

Description

Like the mock data frame "abcd_colour", but with "unique_id" as the "uid".

Usage

```
anxiety
```

Format

`anxiety`**:**

A data frame with 275 rows and 2 columns:

unique_id The unique identifier of the ABCD dataset

cbcl_anxiety_r Ordinal value of impairment on CBCL anxiety, either 0 (no impairment), 1 (borderline clinical), or 2 (clinically impaired)

Source

Though this data is no longer "real" ABCD data, the reference for using ABCD as a data source is below:

Data used in the preparation of this article were obtained from the Adolescent Brain Cognitive DevelopmentSM (ABCD) Study (https://abcdstudy.org), held in the NIMH Data Archive (NDA). This is a multisite, longitudinal study designed to recruit more than 10,000 children age 9-10 and follow them over 10 years into early adulthood. The ABCD Study® is supported by the National Institutes of Health and additional federal partners under award numbers U01DA041048, U01DA050989, U01DA051016, U01DA041022, U01DA051018, U01DA051037, U01DA050987, U01DA041174, U01DA041106, U01DA041117, U01DA041028, U01DA041134, U01DA050988, U01DA051039, U01DA041156, U01DA041025, U01DA041120, U01DA051038, U01DA041148, U01DA041093, U01DA041089, U24DA041123, U24DA041147. A full list of supporters is available at https://abcdstudy.org/federal-partners.html. A listing of participating sites and a complete listing of the study investigators can be found at https://abcdstudy.org/consortium_members/. ABCD consortium investigators designed and implemented the study and/or provided data but did not necessarily participate in the analysis or writing of this report. This manuscript reflects the views of the authors and may not reflect the opinions or views of the NIH or ABCD consortium investigators.

`arrange` *Arrange rows in an object*

Description

Arrange rows in an object

Usage

```
arrange(.data, ...)
```

Arguments

`.data`	The object to arrange columns from.
`...`	Additional arguments for arranging.

Value

Object with arrangeed columns.

`as.data.frame.data_list` *Coerce a `data_list` class object into a `data.frame` class object*

Description

Horizontally joins data frames within a data list into a single data frame, using the `uid` attribute as the joining key.

Usage

```
## S3 method for class 'data_list'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`x`	A `data_list` class object.
`row.names`	Additional parameter passed to `as.data.frame()`.
`optional`	Additional parameter passed to `as.data.frame()`.
`...`	Additional parameter passed to `as.data.frame()`.

Value

dl_df A `data.frame` class object with all the features and observations of `dl`.

as.data.frame.ext_solutions_df

Coerce a `ext_solutions_df` *class object into a* `data.frame` *class object*

Description

Coerce a `ext_solutions_df` class object into a `data.frame` class object

Usage

```
## S3 method for class 'ext_solutions_df'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  keep_attributes = FALSE,
  ...
)
```

Arguments

`x`	A `ext_solutions_df` class object.
`row.names`	Additional parameter passed to `as.data.frame()`.
`optional`	Additional parameter passed to `as.data.frame()`.
`keep_attributes`	If TRUE, resulting data frame includes settings data frame and weights matrix.
`...`	Additional parameter passed to `as.data.frame()`.

Value

A `data.frame` class object with all the columns of x and its contained solutions data frame.

as.data.frame.solutions_df

Coerce a `solutions_df` *class object into a* `data.frame` *class object*

Description

Coerce a `solutions_df` class object into a `data.frame` class object

Usage

```
## S3 method for class 'solutions_df'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  keep_attributes = FALSE,
  ...
)
```

Arguments

`x`	A `solutions_df` class object.
`row.names`	Additional parameter passed to `as.data.frame()`.
`optional`	Additional parameter passed to `as.data.frame()`.
`keep_attributes`	If TRUE, resulting data frame includes settings data frame and weights matrix.
`...`	Additional parameter passed to `as.data.frame()`.

Value

A `data.frame` class object with all the columns of x and its contained solutions data frame.

`as.list.clust_fns_list` *Coerce a* `clust_fns_list` *class object into a* `list` *class object*

Description

Coerce a `clust_fns_list` class object into a `list` class object

Usage

```
## S3 method for class 'clust_fns_list'
as.list(x, ...)
```

Arguments

`x`	A `clust_fns_list` class object.
`...`	Additional parameter passed to `as.list()`.

Value

A `list` class object with all the functions of `x`.

`as.list.data_list` *Coerce a* `data_list` *class object into a* `list` *class object*

Description

Coerce a `data_list` class object into a `list` class object

Usage

```
## S3 method for class 'data_list'
as.list(x, ...)
```

Arguments

`x`	A `data_list` class object.
`...`	Additional parameter passed to `as.list()`.

Value

A `list` class object with all the objects of `x`.

`as.list.dist_fns_list` *Coerce a* `dist_fns_list` *class object into a* `list` *class object*

Description

Coerce a `dist_fns_list` class object into a `list` class object

Usage

```
## S3 method for class 'dist_fns_list'
as.list(x, ...)
```

Arguments

`x`	A `dist_fns_list` class object.
`...`	Additional parameter passed to `as.list()`.

Value

A `list` class object with all the functions of `x`.

as.matrix.weights_matrix

Coerce a weights_matrix class object into a matrix class object

Description

Coerce a weights_matrix class object into a matrix class object

Usage

```
## S3 method for class 'weights_matrix'
as.matrix(x, ...)
```

Arguments

x	A weights_matrix class object.
...	Additional parameter passed to as.matrix().

Value

A matrix and array class object.

assemble_data

Collapse a data frame and/or a data list into a single data frame

Description

Collapse a data frame and/or a data list into a single data frame

Usage

```
assemble_data(data, dl)
```

Arguments

data	A data frame.
dl	A nested list of input data from data_list().

Value

A class "data.frame" object containing all the features of the provided data frame and/or data list.

assoc_pval_heatmap *Heatmap of pairwise associations between features*

Description

Heatmap of pairwise associations between features

Usage

```
assoc_pval_heatmap(
  correlation_matrix,
  scale_diag = "max",
  cluster_rows = TRUE,
  cluster_columns = TRUE,
  show_row_names = TRUE,
  show_column_names = TRUE,
  show_heatmap_legend = FALSE,
  confounders = NULL,
  out_of_models = NULL,
  annotation_colours = NULL,
  labels_colour = NULL,
  split_by_domain = FALSE,
  dl = NULL,
  significance_stars = TRUE,
  slice_font_size = 8,
  ...
)
```

Arguments

`correlation_matrix`
: Matrix containing all pairwise association p-values. The recommended way to obtain this matrix is through the calc_assoc_pval function.

`scale_diag`
: Parameter that controls how the diagonals of the correlation_matrix are adjusted in the heatmap. For best viewing, this is set to "max", which will match the diagonals to whichever pairwise association has the highest p-value.

`cluster_rows`
: Parameter for ComplexHeatmap::Heatmap. Will be ignored if split_by_domain is also provided.

`cluster_columns`
: Parameter for ComplexHeatmap::Heatmap. Will be ignored if split_by_domain is also provided.

`show_row_names`
: Parameter for ComplexHeatmap::Heatmap.

`show_column_names`
: Parameter for ComplexHeatmap::Heatmap.

`show_heatmap_legend`
: Parameter for ComplexHeatmap::Heatmap.

`confounders` A named list where the elements are columns in the correlation_matrix and the names are the corresponding display names.

`out_of_models` Like confounders, but a named list of out of model measures (who are also present as columns in the correlation_matrix).

`annotation_colours`
Named list of heatmap annotations and their colours.

`labels_colour` Vector of colours to use for the columns and rows of the heatmap.

`split_by_domain`
The results of `dl_var_summar` - a data frame that has the domain of every feature in the plotted data. columns of the correlation_matrix. Will be used to "slice" the heatmap into visually separated sections.

`dl` A nested list of input data from `data_list()`.

`significance_stars`
If TRUE (default), plots significance stars on heatmap cells

`slice_font_size`
Font size for domain separating labels.

`...` Additional parameters passed into ComplexHeatmap::Heatmap.

Value

Returns a heatmap (class "Heatmap" from package ComplexHeatmap) that displays the pairwise associations between features from the provided correlation_matrix.

Examples

```
#data_list <- data_list(
#    list(income, "household_income", "demographics", "ordinal"),
#    list(pubertal, "pubertal_status", "demographics", "continuous"),
#    list(fav_colour, "favourite_colour", "demographics", "categorical"),
#    list(anxiety, "anxiety", "behaviour", "ordinal"),
#    list(depress, "depressed", "behaviour", "ordinal"),
#    uid = "unique_id"
#)
#
#assoc_pval_matrix <- calc_assoc_pval_matrix(data_list)
#ap_heatmap <- assoc_pval_heatmap(assoc_pval_matrix)
```

`as_ari_matrix` *Convert an object to an ARI matrix*

Description

This function coerces non-`ari_matrix` class objects into `ari_matrix` class objects.

Usage

```
as_ari_matrix(x)
```

Arguments

`x`	The object to convert into a weights matrix.

Value

An `ari_matrix` class object.

`as_data_list` *Convert an object to a data list*

Description

This function coerces non-`data_list` class objects into `data_list` class objects.

Usage

```
as_data_list(x)
```

Arguments

`x`	The object to convert into a data list.

Value

A `data_list` class object.

`as_settings_df` *Convert an object to a settings data frame*

Description

This function coerces non-`settings_df` class objects into `settings_df` class objects.

Usage

```
as_settings_df(x)
```

Arguments

`x`	The object to convert into a data list.

Value

A `settings_df` class object.

as_sim_mats_list *Convert an object to a similarity matrix list*

Description

This function converts non-`sim_mats_list` class objects into `sim_mats_list` class objects.

Usage

```
as_sim_mats_list(x)
```

Arguments

x	The object to convert into a `sim_mats_list`. Must be a list of square matrices with identical column and row names.

Value

A `sim_mats_list` class object.

as_snf_config *Convert an object to a snf config*

Description

This function coerces non-`snf_config` class objects into `snf_config` class objects.

Usage

```
as_snf_config(x)
```

Arguments

x	The object to convert into a snf config.

Value

A `snf_config` class object.

as_weights_matrix *Convert an object to a weights matrix*

Description

This function converts non-weights_matrix objects into weights_matrix class objects.

Usage

```
as_weights_matrix(x)
```

Arguments

x	The object to convert into a data list.

Value

A weights_matrix class object.

auto_plot *Automatically plot features across clusters*

Description

Given a single row of a solutions data frame and data provided through a data list, this function will return a series of bar and/or jitter plots based on feature types.

Usage

```
auto_plot(
  sol_df_row = NULL,
  dl = NULL,
  cluster_df = NULL,
  return_plots = TRUE,
  save = NULL,
  jitter_width = 6,
  jitter_height = 6,
  bar_width = 6,
  bar_height = 6,
  verbose = FALSE
)
```

Arguments

`sol_df_row` A single row of a solutions data frame.

`dl` A data list containing data to plot.

`cluster_df` Directly provide a cluster_df rather than a solutions matrix. Useful if plotting data from label propagated results.

`return_plots` If `TRUE`, the function will return a list of plots. If FALSE, the function will instead return the full data frame used for plotting.

`save` If a string is provided, plots will be saved and this string will be used to prefix plot names.

`jitter_width` Width of jitter plots if save is specified.

`jitter_height` Height of jitter plots if save is specified.

`bar_width` Width of bar plots if save is specified.

`bar_height` Height of bar plots if save is specified.

`verbose` If TRUE, output progress to console.

Value

By default, returns a list of plots (class "gg", "ggplot") with one plot for every feature in the provided data list and/or target list. If `return_plots` is FALSE, will instead return a single "data.frame" object containing every provided feature for every observation in long format.

`bar_plot` *Bar plot separating a feature by cluster*

Description

Bar plot separating a feature by cluster

Usage

```
bar_plot(df, feature)
```

Arguments

`df` A data.frame containing cluster column and the feature to plot.

`feature` The feature to plot.

Value

A bar plot (class "gg", "ggplot") showing the distribution of a feature across clusters.

`batch_snf` *Run variations of SNF.*

Description

This is the core function of the `metasnf` package. Using the information stored in a settings_df (see ?settings_df) and a data list (see ?data_list), run repeated complete SNF pipelines to generate a broad space of post-SNF cluster solutions.

Usage

```
batch_snf(dl, sc, processes = 1, return_sim_mats = FALSE, sim_mats_dir = NULL)
```

Arguments

`dl` A nested list of input data from `data_list()`.

`sc` An `snf_config` class object which stores all sets of hyperparameters used to transform data in dl into a cluster solutions. See `?settings_df` or https://branchlab.github.io/metasnf/artic for more details.

`processes` Specify number of processes used to complete SNF iterations

- `1` (default) Sequential processing: function will iterate through the `settings_df` one row at a time with a for loop. This option will not make use of multiple CPU cores, but will show a progress bar.
- `2` or higher: Parallel processing will use the `future.apply::future_apply` to distribute the SNF iterations across the specified number of CPU cores. If higher than the number of available cores, a warning will be raised and the maximum number of cores will be used.
- `max`: All available cores will be used.

`return_sim_mats` If TRUE, function will return a list where the first element is the solutions data frame and the second element is a list of similarity matrices for each row in the sol_df. Default FALSE.

`sim_mats_dir` If specified, this directory will be used to save all generated similarity matrices.

Value

By default, returns a solutions data frame (class "data.frame"), a a data frame containing one row for every row of the provided settings matrix, all the original columns of that settings data frame, and new columns containing the assigned cluster of each observation from the cluster solution derived by that row's settings. If `return_sim_mats` is TRUE, the function will instead return a list containing the solutions data frame as well as a list of the final similarity matrices (class "matrix") generated by SNF for each row of the settings data frame. If `suppress_clustering` is TRUE, the solutions data frame will not be returned in the output.

Examples

```
input_dl <- data_list(
    list(gender_df, "gender", "demographics", "categorical"),
    list(diagnosis_df, "diagnosis", "clinical", "categorical"),
    uid = "patient_id"
)

sc <- snf_config(input_dl, n_solutions = 3)

# A solutions data frame without similarity matrices:
sol_df <- batch_snf(input_dl, sc)

# A solutions data frame with similarity matrices:
# sol_df <- batch_snf(input_dl, sc, return_sim_mats = TRUE)
# sim_mats_list(sol_df)
```

`batch_snf_subsamples` *Run SNF clustering pipeline on a list of subsampled data lists.*

Description

Run SNF clustering pipeline on a list of subsampled data lists.

Usage

```
batch_snf_subsamples(
  dl_subsamples,
  sc,
  processes = 1,
  return_sim_mats = FALSE,
  sim_mats_dir = NULL,
  verbose = TRUE
)
```

Arguments

`dl_subsamples` A list of subsampled data lists. This object is generated by the function `batch_snf_subsamples()`.

`sc` An `snf_config` class object which stores all sets of hyperparameters used to transform data in dl into a cluster solutions. See `?settings_df` or https://branchlab.github.io/metasnf/artic for more details.

`processes` Specify number of processes used to complete SNF iterations

- `1` (default) Sequential processing: function will iterate through the `settings_df` one row at a time with a for loop. This option will not make use of multiple CPU cores, but will show a progress bar.
- `2` or higher: Parallel processing will use the `future.apply::future_apply` to distribute the SNF iterations across the specified number of CPU cores. If higher than the number of available cores, a warning will be raised and the maximum number of cores will be used.

- max: All available cores will be used.

return_sim_mats
: If TRUE, function will return a list where the first element is the solutions data frame and the second element is a list of similarity matrices for each row in the sol_df. Default FALSE.

sim_mats_dir
: If specified, this directory will be used to save all generated similarity matrices.

verbose
: If TRUE, output progress to console.

Value

By default, returns a one-element list: cluster_solutions, which is itself a list of cluster solution data frames corresponding to each of the provided data list subsamples. Setting the parameters return_sim_mats and return_solutions to TRUE will turn the result of the function to a three-element list containing the corresponding solutions data frames and final fused similarity matrices of those cluster solutions, should you require these objects for your own stability calculations.

Examples

```
# my_dl <- data_list(
#     list(subc_v, "subcortical_volume", "neuroimaging", "continuous"),
#     list(income, "household_income", "demographics", "continuous"),
#     list(pubertal, "pubertal_status", "demographics", "continuous"),
#     uid = "unique_id"
# )
#
# sc <- snf_config(my_dl, n_solutions = 5, max_k = 40)
#
# my_dl_subsamples <- subsample_dl(
#     my_dl,
#     n_subsamples = 20,
#     subsample_fraction = 0.85
# )
#
# batch_subsample_results <- batch_snf_subsamples(
#     my_dl_subsamples,
#     sc,
#     verbose = TRUE
# )
```

calculate_coclustering

Calculate coclustering data.

Description

Calculate coclustering data.

Usage

```
calculate_coclustering(subsample_solutions, sol_df, verbose = FALSE)
```

Arguments

`subsample_solutions`
: A list of containing cluster solutions from distinct subsamples of the data. This object is generated by the function `batch_snf_subsamples()`. These solutions should correspond to the ones in the solutions data frame.

`sol_df`
: A solutions data frame. This object is generated by the function `batch_snf()`. The solutions in the solutions data frame should correspond to those in the subsample solutions.

`verbose`
: If TRUE, output time remaining estimates to console.

Value

A list containing the following components:

- cocluster_dfs: A list of data frames, one per cluster solution, that shows the number of times that every pair of observations in the original cluster solution occurred in the same subsample, the number of times that every pair clustered together in a subsample, and the corresponding fraction of times that every pair clustered together in a subsample.
- cocluster_ss_mats: The number of times every pair of observations occurred in the same subsample, formatted as a pairwise matrix.
- cocluster_sc_mats: The number of times every pair of observations occurred in the same cluster, formatted as a pairwise matrix.
- cocluster_cf_mats: The fraction of times every pair of observations occurred in the same cluster, formatted as a pairwise matrix.
- cocluster_summary: Specifically among pairs of observations that clustered together in the original full cluster solution, what fraction of those pairs remained clustered together throughout the subsample solutions. This information is formatted as a data frame with one row per cluster solution.

Examples

```
# my_dl <- data_list(
#     list(subc_v, "subcortical_volume", "neuroimaging", "continuous"),
#     list(income, "household_income", "demographics", "continuous"),
#     list(pubertal, "pubertal_status", "demographics", "continuous"),
#     uid = "unique_id"
# )
#
# sc <- snf_config(my_dl, n_solutions = 5, max_k = 40)
#
# sol_df <- batch_snf(my_dl, sc)
#
# my_dl_subsamples <- subsample_dl(
#     my_dl,
#     n_subsamples = 20,
#     subsample_fraction = 0.85
# )
#
# batch_subsample_results <- batch_snf_subsamples(
#     my_dl_subsamples,
#     sc,
#     verbose = TRUE
# )
#
# coclustering_results <- calculate_coclustering(
#     batch_subsample_results,
#     sol_df,
#     verbose = TRUE
# )
```

calc_aris *Construct an ARI matrix storing inter-solution similarities*

Description

This function constructs an `ari_matrix` class object from a `solutions_df` class object. The ARI matrix stores pairwise adjusted Rand indices for all cluster solutions as well as a numeric order for the solutions data frame based on the hierarchical clustering of the ARI matrix.

Usage

```
calc_aris(
  sol_df,
  processes = 1,
  verbose = FALSE,
  dist_method = "euclidean",
  hclust_method = "complete"
)
```

Arguments

`sol_df` Solutions data frame containing cluster solutions to calculate pairwise ARIs for.

`processes` Specify number of processes used to complete calculations

- `1` (default) Sequential processing
- `2` or higher: Parallel processing will use the `future.apply::future_apply` to distribute the calculations across the specified number of CPU cores. If higher than the number of available cores, a warning will be raised and the maximum number of cores will be used.
- `max`: All available cores will be used. Note that no progress indicator is available during multi-core processing.

`verbose` If TRUE, output progress to console.

`dist_method`	Distance method to use when calculating sorting order to of the matrix. Argument is directly passed into stats::dist. Options include "euclidean", "maximum", "manhattan", "canberra", "binary", or "minkowski".
`hclust_method`	Agglomerative method to use when calculating sorting order by `stats::hclust`. Options include "ward.D", "ward.D2", "single", "complete", "average", "mcquitty", "median", or "centroid".

Value

om_aris ARIs between clustering solutions of an solutions data frame

Examples

```
dl <- data_list(
    list(subc_v, "subcortical_volume", "neuroimaging", "continuous"),
    list(pubertal, "pubertal_status", "demographics", "continuous"),
    uid = "unique_id"
)

sc <- snf_config(dl, n_solutions = 3)
sol_df <- batch_snf(dl, sc)
calc_aris(sol_df)
```

---

calc_assoc_pval_matrix

Calculate p-values for all pairwise associations of features in a data list

---

Description

Calculate p-values for all pairwise associations of features in a data list

Usage

```
calc_assoc_pval_matrix(dl, verbose = FALSE, cat_test = "chi_squared")
```

Arguments

`dl`	A nested list of input data from `data_list()`.
`verbose`	If TRUE, output progress to the console.
`cat_test`	String indicating which statistical test will be used to associate cluster with a categorical feature. Options are "chi_squared" for the Chi-squared test and "fisher_exact" for Fisher's exact test.

Value

A "matrix" class object containing pairwise association p-values between the features in the provided data list.

Examples

```
data_list <- data_list(
    list(income, "household_income", "demographics", "ordinal"),
    list(pubertal, "pubertal_status", "demographics", "continuous"),
    list(anxiety, "anxiety", "behaviour", "ordinal"),
    list(depress, "depressed", "behaviour", "ordinal"),
    uid = "unique_id"
)

assoc_pval_matrix <- calc_assoc_pval_matrix(data_list)
```

---

calc_nmis *Calculate feature NMIs for a data list and a solutions data frame*

Description

Normalized mutual information scores can be used to indirectly measure how important a feature may have been in producing a cluster solution. This function will calculate the normalized mutual information between cluster solutions in a solutions data frame as well as cluster solutions created by including only a single feature from a provided data list, but otherwise using all the same hyper-parameters as specified in the original SNF config. Note that NMIs can be calculated between two cluster solutions regardless of what features were actually used to create those cluster solutions. For example, a feature that was not involved in producing a particular cluster solution may still have a high NMI with that cluster solution (typically because it was highly correlated with a different feature that was used).

Usage

```
calc_nmis(
  dl,
  sol_df,
  transpose = TRUE,
  ignore_inclusions = TRUE,
  verbose = FALSE
)
```

Arguments

`dl`	A nested list of input data from `data_list()`.
`sol_df`	Result of `batch_snf` storing cluster solutions and the settings that were used to generate them. Use the same value as was used in the original call to `batch_snf()`.
`transpose`	If TRUE, will transpose the output data frame.
`ignore_inclusions`	If TRUE, will ignore the inclusion columns in the solutions data frame and calculate NMIs for all features. If FALSE, will give NAs for features that were dropped on a given settings_df row.
`verbose`	If TRUE, output progress to console.

Value

A "data.frame" class object containing one row for every feature in the provided data list and one column for every solution in the provided solutions data frame. Populated values show the calculated NMI score for each feature-solution combination.

Examples

```
input_dl <- data_list(
    list(gender_df, "gender", "demographics", "categorical"),
    list(diagnosis_df, "diagnosis", "clinical", "categorical"),
    uid = "patient_id"
)

sc <- snf_config(input_dl, n_solutions = 2)

sol_df <- batch_snf(input_dl, sc)

calc_nmis(input_dl, sol_df)
```

`cancer_diagnosis_df` *Mock diagnosis data*

Description

This is the same data as `diagnosis_df`, with renamed features and columns.

Usage

```
cancer_diagnosis_df
```

Format

`cancer_diagnosis_df`**:**

A data frame with 200 rows and 2 columns:

patient_id Random three-digit number uniquely identifying the patient

diagnosis Mock cancer diagnosis feature (1, 2, or 3)

Source

This data came from the SNFtool package, with slight modifications.

`cell_significance_fn` *Place significance stars on ComplexHeatmap cells.*

Description

This is an internal function meant to be used to by the assoc_pval_heatmap function.

Usage

```
cell_significance_fn(data)
```

Arguments

`data`	The matrix containing the cells to base the significance stars on.

Value

cell_fn Another function that is well-formatted for usage as the cell_fun argument in ComplexHeatmap::Heatmap.

`check_dataless_annotations` *Helper function to stop annotation building when no data was provided*

Description

Helper function to stop annotation building when no data was provided

Usage

```
check_dataless_annotations(annotation_requests, data)
```

Arguments

`annotation_requests`	A list of requested annotations
`data`	A data frame with data to build annotations

Value

Does not return any value. This function just raises an error when annotations are requested without any provided data for a heatmap.

`check_hm_dependencies` *Check for ComplexHeatmap and circlize dependencies*

Description

Check for ComplexHeatmap and circlize dependencies

Usage

```
check_hm_dependencies()
```

Value

Does not return any value. This function just checks that the ComplexHeatmap and circlize packages are installed.

`check_similarity_matrices` *Check validity of similarity matrices*

Description

Check to see if similarity matrices in a list have the following properties:

1. The maximum value in the entire matrix is 0.5
2. Every value in the diagonal is 0.5

Usage

```
check_similarity_matrices(similarity_matrices)
```

Arguments

`similarity_matrices`
A list of similarity matrices

Value

valid_matrices Boolean indicating if properties are met by all similarity matrices `clust_fns` *Built-in clustering algorithms*

Description

These functions can be used when building a `metasnf` clustering functions list. Each function converts a similarity matrix (matrix class object) to a cluster solution (numeric vector). Note that these functions (or custom clustering functions) cannot accept number of clusters as a parameter; this value must be built into the function itself if necessary.

Usage

```
spectral_eigen(similarity_matrix)

spectral_rot(similarity_matrix)

spectral_eigen_classic(similarity_matrix)

spectral_rot_classic(similarity_matrix)

spectral_two(similarity_matrix)

spectral_three(similarity_matrix)

spectral_four(similarity_matrix)

spectral_five(similarity_matrix)

spectral_six(similarity_matrix)

spectral_seven(similarity_matrix)

spectral_eight(similarity_matrix)

spectral_nine(similarity_matrix)

spectral_ten(similarity_matrix)
```

Arguments

`similarity_matrix`
A similarity matrix.

Details

- spectral_eigen: Spectral clustering where the number of clusters is based on the eigen-gap heuristic

- spectral_rot: Spectral clustering where the number of clusters is based on the rotation-cost heuristic
- spectral_(C): Spectral clustering for a C-cluster solution.

Value

solution_data A vector of cluster assignments

clust_fns_list *Build a clusteing algorithms list*

Description

This function can be used to specify custom clustering algorithms to apply to the final similarity matrices produced by each run of the batch_snf function.

Usage

```
clust_fns_list(clust_fns = NULL, use_default_clust_fns = FALSE)
```

Arguments

`clust_fns` A list of named clustering functions

`use_default_clust_fns`
If TRUE, prepend the base clustering algorithms (spectral_eigen and spectral_rot, which apply spectral clustering and use the eigen-gap and rotation cost heuristics respectively for determining the number of clusters in the graph) to clust_fns.

Value

A list of clustering algorithm functions that can be passed into the batch_snf and generate_settings_list functions.

Examples

```
# Using just the base clustering algorithms --------------------------------
# This will just contain spectral_eigen and spectral_rot
cfl <- clust_fns_list(use_default_clust_fns = TRUE)

# Adding algorithms provided by the package --------------------------------
# This will contain the base clustering algorithms (spectral_eigen,
#  spectral_rot) as well as two pre-defined spectral clustering functions
#  that force the number of clusters to be two or five
cfl <- clust_fns_list(
     clust_fns = list(
        "two_cluster_spectral" = spectral_two,
        "five_cluster_spectral" = spectral_five
    )
)

# Adding your own algorithms ---------------------------------------------
# This will contain the base and user-provided clustering algorithms
my_clustering_algorithm <- function(similarity_matrix) {
    # your code that converts similarity matrix to clusters here...
}

# Suppress the base algorithms---------------------------------------------
# This will contain only user-provided clustering algorithms
cfl <- clust_fns_list(
    clust_fns = list(
        "two_cluster_spectral" = spectral_two,
        "five_cluster_spectral" = spectral_five
    )
)
```

cocluster_density *Density plot coclustering stability across subsampled data.*

Description

This function creates a density plot that shows, for all pairs of observations that originally clustered together, the distribution of the the fractions that those pairs clustered together across subsampled data.

Usage

```
cocluster_density(cocluster_df)
```

Arguments

`cocluster_df`	A data frame containing coclustering data for a single cluster solution. This object is generated by the `calculate_coclustering` function.

Value

Density plot (class "gg", "ggplot") of the distribution of coclustering across pairs and subsamples of the data.

Examples

```
# my_dl <- data_list(
#     list(subc_v, "subcortical_volume", "neuroimaging", "continuous"),
#     list(income, "household_income", "demographics", "continuous"),
#     list(pubertal, "pubertal_status", "demographics", "continuous"),
#     uid = "unique_id"
# )
#
```

```
# sc <- snf_config(my_dl, n_solutions = 5, max_k = 40)
#
# sol_df <- batch_snf(my_dl, sc)
#
# my_dl_subsamples <- subsample_dl(
#     my_dl,
#     n_subsamples = 20,
#     subsample_fraction = 0.85
# )
#
# batch_subsample_results <- batch_snf_subsamples(
#     my_dl_subsamples,
#     sc,
#     verbose = TRUE
# )
#
# coclustering_results <- calculate_coclustering(
#     batch_subsample_results,
#     sol_df,
#     verbose = TRUE
# )
#
# cocluster_density(cocluster_dfs[[1]])
```

---

cocluster_heatmap *Heatmap of observation co-clustering across resampled data.*

Description

Create a heatmap that shows the distribution of observation co-clustering across resampled data.

Usage

```
cocluster_heatmap(
  cocluster_df,
  cluster_rows = TRUE,
  cluster_columns = TRUE,
  show_row_names = FALSE,
  show_column_names = FALSE,
  dl = NULL,
  data = NULL,
  left_bar = NULL,
  right_bar = NULL,
  top_bar = NULL,
  bottom_bar = NULL,
  left_hm = NULL,
  right_hm = NULL,
  top_hm = NULL,
  bottom_hm = NULL,
  annotation_colours = NULL,
  min_colour = NULL,
  max_colour = NULL,
  ...
)
```

Arguments

cocluster_df A data frame containing coclustering data for a single cluster solution. This object is generated by the calculate_coclustering function.

cluster_rows Argument passed to ComplexHeatmap::Heatmap().

cluster_columns Argument passed to ComplexHeatmap::Heatmap().

show_row_names Argument passed to ComplexHeatmap::Heatmap().

show_column_names Argument passed to ComplexHeatmap::Heatmap().

dl See ?similarity_matrix_heatmap.

data See ?similarity_matrix_heatmap.

left_bar See ?similarity_matrix_heatmap.

right_bar See ?similarity_matrix_heatmap.

top_bar See ?similarity_matrix_heatmap.

bottom_bar See ?similarity_matrix_heatmap.

left_hm See ?similarity_matrix_heatmap.

right_hm See ?similarity_matrix_heatmap.

top_hm See ?similarity_matrix_heatmap.

bottom_hm See ?similarity_matrix_heatmap.

annotation_colours See ?similarity_matrix_heatmap.

min_colour See ?similarity_matrix_heatmap.

max_colour See ?similarity_matrix_heatmap.

... Arguments passed to ComplexHeatmap::Heatmap().

Value

Heatmap (class "Heatmap" from ComplexHeatmap) object showing the distribution of observation co-clustering across resampled data.

Examples

```
# my_dl <- data_list(
#     list(subc_v, "subcortical_volume", "neuroimaging", "continuous"),
#     list(income, "household_income", "demographics", "continuous"),
#     list(pubertal, "pubertal_status", "demographics", "continuous"),
#     uid = "unique_id"
# )
#
# sc <- snf_config(my_dl, n_solutions = 5, max_k = 40)
#
# sol_df <- batch_snf(my_dl, sc)
#
# my_dl_subsamples <- subsample_dl(
#     my_dl,
#     n_subsamples = 20,
#     subsample_fraction = 0.85
# )
#
# batch_subsample_results <- batch_snf_subsamples(
#     my_dl_subsamples,
#     sc,
#     verbose = TRUE
# )
#
# coclustering_results <- calculate_coclustering(
#     batch_subsample_results,
#     sol_df,
#     verbose = TRUE
# )
#
# cocluster_dfs <- coclustering_results$"cocluster_dfs"
#
# cocluster_heatmap(
#     cocluster_dfs[[1]],
#     dl = my_dl,
#     top_hm = list(
#         "Income" = "household_income",
#         "Pubertal Status" = "pubertal_status"
#     ),
#     annotation_colours = list(
#         "Pubertal Status" = colour_scale(
#             c(1, 4),
#             min_colour = "black",
#             max_colour = "purple"
#         ),
#         "Income" = colour_scale(
#             c(0, 4),
#             min_colour = "black",
#             max_colour = "red"
#         )
#     )
# )
```

collapse_dl *Convert a data list into a data frame*

Description

[Deprecated] Defunct function for converting a data list into a data frame. Please use `as.data.frame()` instead.

Usage

```
collapse_dl(data_list)
```

Arguments

`data_list`	A nested list of input data from `generate_data_list()`.

Value

A "data.frame"-formatted version of the provided data list.

`colour_scale` *Return a colour ramp for a given vector*

Description

Given a numeric vector and min and max colour values, return a colour ramp that assigns a colour to each element in the vector. This function is a wrapper for `circlize::colorRamp2`.'

Usage

```
colour_scale(data, min_colour, max_colour)
```

Arguments

`data`	Vector of numeric values.
`min_colour`	Minimum colour value.
`max_colour`	Maximum colour value.

Value

A "function" class object that can build a circlize-style colour ramp.

config_heatmap *Heatmap for visualizing an SNF config*

Description

Create a heatmap where each row corresponds to a different set of hyperparameters in an SNF config object. Numeric parameters are scaled normalized and non-numeric parameters are added as heatmap annotations. Rows can be reordered to match prior meta clustering results.

Usage

```
config_heatmap(
  sc,
  order = NULL,
  hide_fixed = FALSE,
  show_column_names = TRUE,
  show_row_names = TRUE,
  rect_gp = grid::gpar(col = "black"),
  colour_breaks = c(0, 1),
  colours = c("black", "darkseagreen"),
  column_split_vector = NULL,
  row_split_vector = NULL,
  column_split = NULL,
  row_split = NULL,
  column_title = NULL,
  include_weights = TRUE,
  include_settings = TRUE,
  ...
)
```

Arguments

`sc` An `snf_config` class object.

`order` Numeric vector indicating row ordering of SNF config.

`hide_fixed` Whether fixed parameters should be removed.

`show_column_names`
Whether show column names.

`show_row_names` Whether show row names.

`rect_gp` Graphic parameters for drawing rectangles (for heatmap body). The value should be specified by `gpar` and `fill` parameter is ignored.

`colour_breaks` Numeric vector of breaks for the legend.

`colours` Vector of colours to use for the heatmap. Should match the length of colour_breaks.

`column_split_vector`
Vector of indices to split columns by.

`row_split_vector`
Vector of indices to split rows by.

`column_split` Split on columns. For heatmap splitting, please refer to https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#heatmap-split .

`row_split` Same as `split`.

`column_title` Title on the column.

`include_weights`
If TRUE, includes feature weights of the weights matrix into the config heatmap.

`include_settings`
If TRUE, includes columns from the settings data frame into the config heatmap.

`...` Additional parameters passed to `ComplexHeatmap::Heatmap`.

Value

Returns a heatmap (class "Heatmap" from package ComplexHeatmap) that displays the scaled values of the provided SNF config.

Examples

```
dl <- data_list(
    list(income, "household_income", "demographics", "ordinal"),
    list(pubertal, "pubertal_status", "demographics", "continuous"),
    list(fav_colour, "favourite_colour", "demographics", "categorical"),
    list(anxiety, "anxiety", "behaviour", "ordinal"),
    list(depress, "depressed", "behaviour", "ordinal"),
    uid = "unique_id"
)

sc <- snf_config(
    dl,
    n_solutions = 10,
    dropout_dist = "uniform"
)

config_heatmap(sc)
```

`cort_sa` *Mock ABCD cortical surface area data*

Description

Like the mock data frame "abcd_cort_sa", but with "unique_id" as the "uid".

Usage

```
cort_sa
```

Format

`cort_sa`**:**

A data frame with 188 rows and 152 columns:

unique_id The unique identifier of the ABCD dataset

... Cortical surface areas of various ROIs (mm^2, I think)

Source

Though this data is no longer "real" ABCD data, the reference for using ABCD as a data source is below:

Data used in the preparation of this article were obtained from the Adolescent Brain Cognitive DevelopmentSM (ABCD) Study (https://abcdstudy.org), held in the NIMH Data Archive (NDA). This is a multisite, longitudinal study designed to recruit more than 10,000 children age 9-10 and follow them over 10 years into early adulthood. The ABCD Study® is supported by the National Institutes of Health and additional federal partners under award numbers U01DA041048, U01DA050989, U01DA051016, U01DA041022, U01DA051018, U01DA051037, U01DA050987, U01DA041174, U01DA041106, U01DA041117, U01DA041028, U01DA041134, U01DA050988, U01DA051039, U01DA041156, U01DA041025, U01DA041120, U01DA051038, U01DA041148, U01DA041093, U01DA041089, U24DA041123, U24DA041147. A full list of supporters is available at https://abcdstudy.org/federal-partners.html. A listing of participating sites and a complete listing of the study investigators can be found at https://abcdstudy.org/consortium_members/. ABCD consortium investigators designed and implemented the study and/or provided data but did not necessarily participate in the analysis or writing of this report. This manuscript reflects the views of the authors and may not reflect the opinions or views of the NIH or ABCD consortium investigators.

`cort_t` *Mock ABCD cortical thickness data*

Description

Like the mock data frame "abcd_cort_t", but with "unique_id" as the "uid".

Usage

```
cort_t
```

Format

`cort_t`**:**

A data frame with 188 rows and 152 columns:

unique_id The unique identifier of the ABCD dataset

... Cortical thicknesses of various ROIs (mm^3, I think)

Source

Though this data is no longer "real" ABCD data, the reference for using ABCD as a data source is below:

Data used in the preparation of this article were obtained from the Adolescent Brain Cognitive DevelopmentSM (ABCD) Study (https://abcdstudy.org), held in the NIMH Data Archive (NDA). This is a multisite, longitudinal study designed to recruit more than 10,000 children age 9-10 and follow them over 10 years into early adulthood. The ABCD Study® is supported by the National Institutes of Health and additional federal partners under award numbers U01DA041048, U01DA050989, U01DA051016, U01DA041022, U01DA051018, U01DA051037, U01DA050987, U01DA041174, U01DA041106, U01DA041117, U01DA041028, U01DA041134, U01DA050988, U01DA051039, U01DA041156, U01DA041025, U01DA041120, U01DA051038, U01DA041148, U01DA041093, U01DA041089, U24DA041123, U24DA041147. A full list of supporters is available at https://abcdstudy.org/federal-partners.html. A listing of participating sites and a complete listing of the study investigators can be found at https://abcdstudy.org/consortium_members/. ABCD consortium investigators designed and implemented the study and/or provided data but did not necessarily participate in the analysis or writing of this report. This manuscript reflects the views of the authors and may not reflect the opinions or views of the NIH or ABCD consortium investigators.

data_list *Build a* data_list *class object*

Description

data_list() constructs a data list object which inherits from classes data_list and list. This object is the primary way in which features to be used along the metasnf clustering pipeline are stored. The data list is fundamentally a 2-level nested list object where each inner list contains a data frame and associated metadata for that data frame. The metadata includes the name of the data frame, the 'domain' of that data frame (the broader source of information that the input data frame is capturing, determined by user's domain knowledge), and the type of feature stored in the data frame (continuous, discrete, ordinal, categorical, or mixed).

Usage

```
data_list(..., uid)
```

Arguments

...	Any number of lists formatted as (df, "df_name", "df_domain", "df_type") and/or any number of lists of lists formatted as (df, "df_name", "df_domain", "df_type").
uid	(character) the name of the uid column currently used data. data frame.

Examples

```
heart_rate_df <- data.frame(
    patient_id = c("1", "2", "3"),
    var1 = c(0.04, 0.1, 0.3),
    var2 = c(30, 2, 0.3)
)

personality_test_df <- data.frame(
    patient_id = c("1", "2", "3"),
    var3 = c(900, 1990, 373),
    var4 = c(509, 2209, 83)
)

survey_response_df <- data.frame(
    patient_id = c("1", "2", "3"),
    var5 = c(1, 3, 3),
    var6 = c(2, 3, 3)
)

city_df <- data.frame(
    patient_id = c("1", "2", "3"),
    var7 = c("toronto", "montreal", "vancouver")
)

# Explicitly (Name each nested list element):
dl <- data_list(
    list(
        data = heart_rate_df,
        name = "heart_rate",
        domain = "clinical",
        type = "continuous"
    ),
    list(
        data = personality_test_df,
        name = "personality_test",
        domain = "surveys",
        type = "continuous"
    ),
    list(
        data = survey_response_df,
        name = "survey_response",
        domain = "surveys",
        type = "ordinal"
    ),
    list(
        data = city_df,
        name = "city",
        domain = "location",
        type = "categorical"
    ),
    uid = "patient_id"
)

# Compact loading
dl <- data_list(
    list(heart_rate_df, "heart_rate", "clinical", "continuous"),
    list(personality_test_df, "personality_test", "surveys", "continuous"),
    list(survey_response_df, "survey_response", "surveys", "ordinal"),
    list(city_df, "city", "location", "categorical"),
    uid = "patient_id"
)

# Printing data list summaries
summary(dl)

# Alternative loading: providing a single list of lists
list_of_lists <- list(
    list(heart_rate_df, "data1", "domain1", "continuous"),
    list(personality_test_df, "data2", "domain2", "continuous")
)

dl <- data_list(
    list_of_lists,
    uid = "patient_id"
)
```

`depress` *Mock ABCD depression data*

Description

Like the mock data frame "abcd_depress", but with "unique_id" as the "uid".

Usage

```
depress
```

Format

`depress`**:**

A data frame with 275 rows and 2 columns:

unique_id The unique identifier of the ABCD dataset

cbcl_depress_r Ordinal value of impairment on CBCL anxiety, either 0 (no impairment), 1 (borderline clinical), or 2 (clinically impaired)

Source

Though this data is no longer "real" ABCD data, the reference for using ABCD as a data source is below:

Data used in the preparation of this article were obtained from the Adolescent Brain Cognitive DevelopmentSM (ABCD) Study (https://abcdstudy.org), held in the NIMH Data Archive (NDA). This is a multisite, longitudinal study designed to recruit more than 10,000 children age 9-10 and follow them over 10 years into early adulthood. The ABCD Study® is supported by the National Institutes of Health and additional federal partners under award numbers U01DA041048, U01DA050989, U01DA051016, U01DA041022, U01DA051018, U01DA051037, U01DA050987,

U01DA041174, U01DA041106, U01DA041117, U01DA041028, U01DA041134, U01DA050988, U01DA051039, U01DA041156, U01DA041025, U01DA041120, U01DA051038, U01DA041148, U01DA041093, U01DA041089, U24DA041123, U24DA041147. A full list of supporters is available at https://abcdstudy.org/federal-partners.html. A listing of participating sites and a complete listing of the study investigators can be found at https://abcdstudy.org/consortium_members/. ABCD consortium investigators designed and implemented the study and/or provided data but did not necessarily participate in the analysis or writing of this report. This manuscript reflects the views of the authors and may not reflect the opinions or views of the NIH or ABCD consortium investigators.

diagnosis_df *Mock diagnosis data*

Description

This is the same data as `cancer_diagnosis_df`, with renamed features and columns.

Usage

```
diagnosis_df
```

Format

`diagnosis_df`**:**

A data frame with 200 rows and 2 columns:

patient_id Random three-digit number uniquely identifying the patient

diagnosis Mock diagnosis feature

Source

This data came from the SNFtool package, with slight modifications.

dist_fns *Built-in distance functions*

Description

These functions can be used when building a `metasnf` distance functions list. Each function converts a data frame into to a distance matrix.

Usage

```
euclidean_distance(df, weights_row)

gower_distance(df, weights_row)

sn_euclidean_distance(df, weights_row)

sew_euclidean_distance(df, weights_row)

hamming_distance(df, weights_row)
```

Arguments

`df`	Data frame containing at least 1 data column
`weights_row`	Single-row data frame where the column names contain the column names in df and the row contains the corresponding weights_row.

Details

Functions that work for numeric data:

- euclidean_distance: typical Euclidean distance
- sn_euclidean_distance: Data frame is first standardized and normalized before typical Euclidean distance is applied
- siw_euclidean_distance: Squared (including weights) Euclidean distance, where the weights are also squared
- sew_euclidean_distance: Squared (excluding weights) Euclidean distance, where the weights are not also squared

Functions that work for binary data:

- hamming_distance: typical Hamming distance

Functions that work for any type of data:

- gower_distance: Gower distance (cluster::daisy)

Value

A matrix class object containing pairwise distances.

dist_fns_list *Build a distance metrics list*

Description

The distance metrics list object (inherits classes `dist_fns_list` and `list`) is a list that stores R functions which can convert a data frame of features into a matrix of pairwise distances. The list is a nested one, where the first layer of the list can hold up to 5 items (one for each of the `metasnf` recognized feature types, continuous, discrete, ordinal, categorical, and mixed), and the second layer can hold an arbitrary number of distance functions for each of those types.

Usage

```
dist_fns_list(
  cnt_dist_fns = NULL,
  dsc_dist_fns = NULL,
  ord_dist_fns = NULL,
  cat_dist_fns = NULL,
  mix_dist_fns = NULL,
  automatic_standard_normalize = FALSE,
  use_default_dist_fns = FALSE
)
```

Arguments

`cnt_dist_fns` A named list of continuous distance metric functions.

`dsc_dist_fns` A named list of discrete distance metric functions.

`ord_dist_fns` A named list of ordinal distance metric functions.

`cat_dist_fns` A named list of categorical distance metric functions.

`mix_dist_fns` A named list of mixed distance metric functions.

`automatic_standard_normalize`
If TRUE, will automatically use standard normalization prior to calculation of any numeric distances. This parameter overrides all other distance functions list-related parameters.

`use_default_dist_fns`
If TRUE, prepend the base distance metrics (euclidean distance for continuous, discrete, and ordinal data and gower distance for categorical and mixed data) to the resulting distance metrics list.

Details

Call ?distance_metrics to see all distance metric functions provided in metasnf.

Value

A distance metrics list object.

Examples

```
# Using just the base distance metrics  ------------------------------------
dist_fns_list <- dist_fns_list()

# Adding your own metrics ---------------------------------------------------
# This will contain only the and user-provided distance function:
cubed_euclidean <- function(df, weights_row) {
    # (your code that converts a data frame to a distance metric here...)
    weights <- diag(weights_row, nrow = length(weights_row))
    weighted_df <- as.matrix(df) %*% weights
    distance_matrix <- weighted_df |>
        stats::dist(method = "euclidean") |>
        as.matrix()
    distance_matrix <- distance_matrix^3
    return(distance_matrix)
}

dist_fns_list <- dist_fns_list(
    cnt_dist_fns = list(
         "my_cubed_euclidean" = cubed_euclidean
    )
)

# Using default base metrics------------------------------------------------
# Call ?distance_metrics to see all distance metric functions provided in
# metasnf. The code below will contain a mix of user-provided and built-in
# distance metric functions.
dist_fns_list <- dist_fns_list(
    cnt_dist_fns = list(
         "my_distance_metric" = cubed_euclidean
    ),
    dsc_dist_fns = list(
         "my_distance_metric" = cubed_euclidean
    ),
    ord_dist_fns = list(
         "my_distance_metric" = cubed_euclidean
    ),
    cat_dist_fns = list(
         "my_distance_metric" = gower_distance
    ),
    mix_dist_fns = list(
         "my_distance_metric" = gower_distance
    ),
    use_default_dist_fns = TRUE
)
```

dlapply *Lapply-like function for data list objects*

Description

This function enables manipulating a `data_list` class object with lapply syntax without removing that object's `data_list` class attribute. The function will only preserve this attribute if the result of the apply call has a valid data list structure.

Usage

```
dlapply(X, FUN, ...)
```

Arguments

`X`	A `data_list` class object.
`FUN`	The function to be applied to each data list component.
`...`	Optional arguments to `FUN`.

Value

If FUN applied to each component of X yields a valid data list, a data list. Otherwise, a list.

Examples

```
# Convert all UID values to lowercase
dl <- data_list(
    list(abcd_income, "income", "demographics", "discrete"),
    list(abcd_colour, "colour", "likes", "categorical"),
    uid = "patient"
)

dl_lower <- dlapply(
    dl,
    function(x) {
        x$"data"$"uid" <- tolower(x$"data"$"uid")
        return(x)
    }
)
```

`dl_variable_summary` *Variable-level summary of a data list*

Description

[Deprecated] Defunct function to summarize a data list. Please use `summary()` with argument `scope = "feature"` instead.

Usage

```
dl_variable_summary(dl)
```

Arguments

`dl`	A nested list of input data from `data_list()`.

Value

variable_level_summary A data frame containing the name, type, and domain of every variable in a data list.

`dplyr_row_slice.ext_solutions_df`
Function to extend dplyr to extended solutions data frame objects

---

Description

Function to extend dplyr to extended solutions data frame objects

Usage

```
dplyr_row_slice.ext_solutions_df(data, i, ...)
```

Arguments

`data`	An extended solutions data frame.
`i`	A vector of row indices.
`...`	Additional arguments.

Value

Row sliced object with appropriately preserved attributes.

`dplyr_row_slice.solutions_df`
Function to extend dplyr to solutions data frame objects

Description

Function to extend dplyr to solutions data frame objects

Usage

```
dplyr_row_slice.solutions_df(data, i, ...)
```

Arguments

`data`	A solutions data frame.
`i`	A vector of row indices.
`...`	Additional arguments.

Value

Row sliced object with appropriately preserved attributes.

`esm_manhattan_plot` *Manhattan plot of feature-cluster association p-values*

Description

Manhattan plot of feature-cluster association p-values

Usage

```
esm_manhattan_plot(
  esm,
  neg_log_pval_thresh = 5,
  threshold = NULL,
  point_size = 5,
  jitter_width = 0.1,
  jitter_height = 0.1,
  text_size = 15,
  plot_title = NULL,
  hide_x_labels = FALSE,
  bonferroni_line = FALSE
)
```

Arguments

`esm`	Extended solutions data frame storing associations between features and cluster assignments. See `?extend_solutions`.
`neg_log_pval_thresh`	Threshold for negative log p-values.
`threshold`	P-value threshold to plot dashed line at.
`point_size`	Size of points in the plot.
`jitter_width`	Width of jitter.
`jitter_height`	Height of jitter.
`text_size`	Size of text in the plot.
`plot_title`	Title of the plot.
`hide_x_labels`	If TRUE, hides x-axis labels.
`bonferroni_line`	If TRUE, plots a dashed black line at the Bonferroni-corrected equivalent of the p-value threshold.

Value

A Manhattan plot (class "gg", "ggplot") showing the association p-values of features against each solution in the provided solutions data frame.

Examples

```
# full_dl <- data_list(
#     list(subc_v, "subcortical_volume", "neuroimaging", "continuous"),
#     list(income, "household_income", "demographics", "continuous"),
#     list(pubertal, "pubertal_status", "demographics", "continuous"),
#     list(anxiety, "anxiety", "behaviour", "ordinal"),
#     list(depress, "depressed", "behaviour", "ordinal"),
#     uid = "unique_id"
# )
#
# dl <- full_dl[1:3]
# target_dl <- full_dl[4:5]
#
# set.seed(42)
# sc <- snf_config(
#     dl = dl,
#     n_solutions = 20,
#     min_k = 20,
#     max_k = 50
# )
#
# sol_df <- batch_snf(dl, sc)
#
# ext_sol_df <- extend_solutions(
#     sol_df,
#     dl = dl,
#     target = target_dl,
#     min_pval = 1e-10 # p-values below 1e-10 will be thresholded to 1e-10
# )
#
# esm_manhattan <- esm_manhattan_plot(
#     ext_sol_df[1:5, ],
#     neg_log_pval_thresh = 5,
#     threshold = 0.05,
#     point_size = 3,
#     jitter_width = 0.1,
#     jitter_height = 0.1,
#     plot_title = "Feature-Solution Associations",
#     text_size = 14,
#     bonferroni_line = TRUE
# )
```

estimate_nclust_given_graph

Estimate number of clusters for a similarity matrix

Description

Calculate eigengap and rotation-cost estimates of the number of clusters to use when clustering a similarity matrix. This function was adapted from `SNFtool::estimateClustersGivenGraph`, but scales up the Laplacian operator prior to eigenvalue calculations to minimize the risk of floating point-related errors.

Usage

```
estimate_nclust_given_graph(W, NUMC = 2:10)
```

Arguments

`W` Similarity matrix to calculate number of clusters for.

`NUMC` Range of cluster counts to consider among when picking best number of clusters.

Value

A list containing the top two eigengap and rotation-cost estimates for the number of clusters in a given similarity matrix.

Examples

```
input_dl <- data_list(
    list(gender_df, "gender", "demographics", "categorical"),
    list(diagnosis_df, "diagnosis", "clinical", "categorical"),
    uid = "patient_id"
)

sc <- snf_config(input_dl, n_solutions = 1)
sol_df <- batch_snf(input_dl, sc, return_sim_mats = TRUE)
sim_mat <- sim_mats_list(sol_df)[[1]]
estimate_nclust_given_graph(sim_mat)
```

`expression_df` *Modification of SNFtool mock data frame "Data1"*

Description

Modification of SNFtool mock data frame "Data1"

Usage

```
expression_df
```

Format

`expression_df`:

A data frame with 200 rows and 3 columns:

gene_1_expression Mock gene expression feature

gene_2_expression Mock gene expression feature

patient_id Random three-digit number uniquely identifying the patient

Source

This data came from the SNFtool package, with slight modifications.

`extend_solutions` *Extend a solutions data frame to include outcome evaluations*

Description

Extend a solutions data frame to include outcome evaluations

Usage

```
extend_solutions(
  sol_df,
  target_dl = NULL,
  dl = NULL,
  cat_test = "chi_squared",
  min_pval = 1e-10,
  processes = 1,
  verbose = FALSE
)
```

Arguments

`sol_df`	Result of `batch_snf` storing cluster solutions and the settings that were used to generate them.
`target_dl`	A data list with features to calculate p-values for. Features in the target list will be included during p-value summary measure calculations.
`dl`	A data list with features to calcualte p-values for, but that should not be incorporated into p-value summary measure columns (i.e., min/mean/max p-value columns).
`cat_test`	String indicating which statistical test will be used to associate cluster with a categorical feature. Options are "chi_squared" for the Chi-squared test and "fisher_exact" for Fisher's exact test.
`min_pval`	If assigned a value, any p-value less than this will be replaced with this value.
`processes`	The number of processes to use for parallelization. Progress is only reported for sequential processing (processes = 1).
`verbose`	If TRUE, output progress to console.

Value

An extended solutions data frame (`ext_sol_df` class object) that contains p-value columns for each outcome in the provided data lists

Examples

```
input_dl <- data_list(
    list(gender_df, "gender", "demographics", "categorical"),
    list(diagnosis_df, "diagnosis", "clinical", "categorical"),
    uid = "patient_id"
)

sc <- snf_config(input_dl, n_solutions = 2)

sol_df <- batch_snf(input_dl, sc)

ext_sol_df <- extend_solutions(sol_df, input_dl)
```

`fav_colour` *Mock ABCD "colour" data*

Description

Like the mock data frame "abcd_colour", but with "unique_id" as the "uid".

Usage

```
fav_colour
```

Format

`fav_colour`**:**

A data frame with 275 rows and 2 columns:

unique_id The unique identifier of the ABCD dataset

colour Categorical transformation of `cbcl_depress`.

Source

Though this data is no longer "real" ABCD data, the reference for using ABCD as a data source is below:

Data used in the preparation of this article were obtained from the Adolescent Brain Cognitive DevelopmentSM (ABCD) Study (https://abcdstudy.org), held in the NIMH Data Archive (NDA). This is a multisite, longitudinal study designed to recruit more than 10,000 children age 9-10 and follow them over 10 years into early adulthood. The ABCD Study® is supported by the National Institutes of Health and additional federal partners under award numbers U01DA041048, U01DA050989, U01DA051016, U01DA041022, U01DA051018, U01DA051037, U01DA050987, U01DA041174, U01DA041106, U01DA041117, U01DA041028, U01DA041134, U01DA050988,

U01DA051039, U01DA041156, U01DA041025, U01DA041120, U01DA051038, U01DA041148, U01DA041093, U01DA041089, U24DA041123, U24DA041147. A full list of supporters is available at https://abcdstudy.org/federal-partners.html. A listing of participating sites and a complete listing of the study investigators can be found at https://abcdstudy.org/consortium_members/. ABCD consortium investigators designed and implemented the study and/or provided data but did not necessarily participate in the analysis or writing of this report. This manuscript reflects the views of the authors and may not reflect the opinions or views of the NIH or ABCD consortium investigators.

`features` *Return character vector of features stored in an object*

Description

Return character vector of features stored in an object

Usage

```
features(x)
```

Arguments

`x`	The object to pull features from.

Value

A character vector of features in x.

`gender_df` *Mock gender data*

Description

Mock gender data

Usage

```
gender_df
```

Format

`gender_df`**:**
A data frame with 200 rows and 2 columns:

patient_id Random three-digit number uniquely identifying the patient

gender_df Mock gene methylation feature

Source

This data came from the SNFtool package, with slight modifications.

generate_clust_algs_list

Generate a clustering algorithms list

Description

[Deprecated] Deprecated function for building a clustering algorithms list. Please use `clust_fns_list()` (or better yet, `snf_config()`) instead.

Usage

```
generate_clust_algs_list(..., disable_base = FALSE)
```

Arguments

`...`	An arbitrary number of named clustering functions
`disable_base`	If TRUE, do not prepend the base clustering algorithms (spectral_eigen and spectral_rot, which apply spectral clustering and use the eigen-gap and rotation cost heuristics respectively for determining the number of clusters in the graph.

Value

A list of clustering algorithm functions that can be passed into the batch_snf and generate_settings_list functions.

generate_distance_metrics_list

Generate a list of distance metrics

Description

[Deprecated] Deprecated function for building a distance metrics list. Please use `dist_fns_list()` (or better yet, `snf_config()`) instead.

Usage

```
generate_distance_metrics_list(
  continuous_distances = NULL,
  discrete_distances = NULL,
  ordinal_distances = NULL,
  categorical_distances = NULL,
  mixed_distances = NULL,
  keep_defaults = TRUE
)
```

Arguments

`continuous_distances`
: A named list of distance metric functions

`discrete_distances`
: A named list of distance metric functions

`ordinal_distances`
: A named list of distance metric functions

`categorical_distances`
: A named list of distance metric functions

`mixed_distances`
: A named list of distance metric functions

`keep_defaults`
: If TRUE (default), prepend the base distance metrics (euclidean and standard normalized euclidean)

Value

A nested and named list of distance metrics functions.

`generate_settings_matrix` *Build a settings data frame*

Description

[Deprecated] Deprecated function for building a settings matrix. Please use `settings_df()` instead.

Usage

```
generate_settings_matrix(...)
```

Arguments

`...`
: Arguments used to generate a settings matrix.

Value

Raises a deprecated error.

`get_clusters` *Extract cluster membership vector from one solutions data frame row*

Description

[Deprecated] Deprecated function for building extracting cluster solutions from a solutions data frame. Please use `t()` instead.

This function takes in a single row of a solutions data frame and returns a vector containing the cluster assignments for each observation. It is similar to `get_cluster_df()`, which takes a solutions data frame with only one row and returns a data frame with two columns: "cluster" and "uid" '(the UID of the observation) and `get_cluster_solutions()`, which takes a solutions data frame with any number of rows and returns a data frame indicating the cluster assignments for each of those rows.

Usage

```
get_clusters(sol_df_row)
```

Arguments

`sol_df_row`	Output matrix row.

Value

clusters Vector of assigned clusters.

`get_cluster_df` *Extract cluster membership information from one solutions data frame row*

Description

[Deprecated] Deprecated function for building extracting cluster solutions from a solutions data frame. Please use `t()` instead.

This function takes in a single row of a solutions data frame and returns a data frame containing the cluster assignments for each uid. It is similar to `get_clusters()`, which takes one solutions data frame row and returns a vector of cluster assignments' and `get_cluster_solutions()`, which takes a solutions data frame with any number of rows and returns a data frame indicating the cluster assignments for each of those rows.

Usage

```
get_cluster_df(sol_df_row)
```

Arguments

`sol_df_row`	One row from a solutions data frame.

Value

cluster_df data frame of cluster and uid.

`get_cluster_solutions` *Extract cluster membership information from a sol_df*

Description

[Deprecated] Deprecated function for building extracting cluster solutions from a solutions data frame. Please use `t()` instead.

This function takes in a solutions data frame and returns a data frame containing the cluster assignments for each uid. It is similar to '`get_clusters()`, which takes one solutions data frame row and returns a vector of cluster assignments' and `get_cluster_df()`, which takes a solutions matrix with only one row and returns a data frame with two columns: "cluster" and "uid" (the UID of the observation).

Usage

```
get_cluster_solutions(sol_df)
```

Arguments

`sol_df`	A sol_df.

Value

A "data.frame" object where each row is an observation and each column (apart from the uid column) indicates the cluster that observation was assigned to for the corresponding solutions data frame row.

`get_complete_uids` *Pull complete-data UIDs from a list of data frames*

Description

This function identifies all observations within a list of data frames that have no missing data across all data frames. This function is useful when constructing data lists of distinct feature sets from the same sample of observations. As `data_list()` strips away observations with any missing data, distinct sets of observations may be generated by building a data list from the same group of observations over different sets of features. Reducing the pool of observations to only those with complete UIDs first will avoid downstream generation of data lists of differing sizes.

Usage

```
get_complete_uids(list_of_dfs, uid)
```

Arguments

`list_of_dfs`	List of data frames.
`uid`	Name of column across data frames containing UIDs

Value

A character vector of the UIDs of observations that have complete data across the provided list of data frames.

Examples

```
complete_uids <- get_complete_uids(
    list(income, pubertal, anxiety, depress),
    uid = "unique_id"
)

income <- income[income$"unique_id" %in% complete_uids, ]
pubertal <- pubertal[pubertal$"unique_id" %in% complete_uids, ]
anxiety <- anxiety[anxiety$"unique_id" %in% complete_uids, ]
depress <- depress[depress$"unique_id" %in% complete_uids, ]

input_dl <- data_list(
    list(income, "income", "demographics", "ordinal"),
    list(pubertal, "pubertal", "demographics", "continuous"),
    uid = "unique_id"
)

target_dl <- data_list(
    list(anxiety, "anxiety", "behaviour", "ordinal"),
    list(depress, "depressed", "behaviour", "ordinal"),
    uid = "unique_id"
)
```

`get_dl_uids` *Extract UIDs from a data list*

Description

[Deprecated] Deprecated function for extracting UIDs from a data list. Please use `uids()` instead.

Usage

```
get_dl_uids(dl, prefix = FALSE)
```

Arguments

`dl`	A nested list of input data from `data_list()`.
`prefix`	If TRUE, preserves the "uid_" prefix added to UIDs when creating a data list.

Value

A character vector of the UID labels contained in a data list.

`get_heatmap_order` *Return the row or column ordering present in a heatmap*

Description

Return the row or column ordering present in a heatmap

Usage

```
get_heatmap_order(heatmap, type = "rows")
```

Arguments

`heatmap`	A heatmap object to collect ordering from.
`type`	The type of ordering to return. Either "rows" or "columns".

Value

A numeric vector of the ordering used within the provided ComplexHeatmap "Heatmap" object.

`get_matrix_order` *Return the hierarchical clustering order of a matrix*

Description

Return the hierarchical clustering order of a matrix

Usage

```
get_matrix_order(matrix, dist_method = "euclidean", hclust_method = "complete")
```

Arguments

`matrix` Matrix to cluster.

`dist_method` Distance method to use when calculating sorting order to of the matrix. Argument is directly passed into stats::dist. Options include "euclidean", "maximum", "manhattan", "canberra", "binary", or "minkowski".

`hclust_method` Agglomerative method to use when calculating sorting order by `stats::hclust`. Options include "ward.D", "ward.D2", "single", "complete", "average", "mcquitty", "median", or "centroid".

Value

A numeric vector of the ordering derivied by the specified hierarchical clustering method applied to the provided matrix.

Examples

```
# dl <- data_list(
#     list(subc_v, "subcortical_volume", "neuroimaging", "continuous"),
#     list(income, "household_income", "demographics", "continuous"),
#     list(pubertal, "pubertal_status", "demographics", "continuous"),
#     list(anxiety, "anxiety", "behaviour", "ordinal"),
#     list(depress, "depressed", "behaviour", "ordinal"),
#     uid = "unique_id"
# )
#
# sc <- snf_config(
#     dl = dl,
#     n_solutions = 20,
#     min_k = 20,
#     max_k = 50
# )
#
# sol_df <- batch_snf(dl, sc)
#
# ext_sol_df <- extend_solutions(
#     sol_df,
#     dl = dl,
#     min_pval = 1e-10 # p-values below 1e-10 will be thresholded to 1e-10
# )
#
# # Calculate pairwise similarities between cluster solutions
# sol_aris <- calc_aris(sol_df)
#
# # Extract hierarchical clustering order of the cluster solutions
# meta_cluster_order <- get_matrix_order(sol_aris)
```

get_pvals *Get p-values from an extended solutions data frame*

Description

This function can be used to neatly format the p-values associated with an extended solutions data frame. It can also calculate the negative logs of those p-values to make it easier to interpret large-scale differences.

Usage

```
get_pvals(ext_sol_df, negative_log = FALSE, keep_summaries = TRUE)
```

Arguments

ext_sol_df	The output of extend_solutions. A data frame that contains at least one p-value column ending in "_pval".
negative_log	If TRUE, will replace p-values with negative log p-values.
keep_summaries	If FALSE, will remove the mean, min, and max p-value.

Value

A "data.frame" class object Of only the p-value related columns of the provided ext_sol_df.

get_representative_solutions *Extract representative solutions from a matrix of ARIs*

Description

Following clustering with batch_snf, a matrix of pairwise ARIs that show how related each cluster solution is to each other can be generated by the calc_aris function. Partitioning of the ARI matrix can be done by visual inspection of meta_cluster_heatmap() results or by shiny_annotator. Given the indices of meta cluster boundaries, this function will return a single representative solution from each meta cluster based on maximum average ARI to all other solutions within that meta cluster.

Usage

```
get_representative_solutions(aris, sol_df, filter_fn = NULL)
```

Arguments

`aris` Matrix of adjusted rand indices from `calc_aris()`

`sol_df` Output of `batch_snf` containing cluster solutions.

`filter_fn` Optional function to filter the meta-cluster by prior to maximum average ARI determination. This can be useful if you are explicitly trying to select a solution that meets a certain condition, such as only picking from the 4 cluster solutions within a meta cluster. An example valid function could be `fn <- function(x) x[x$"nclust" == 4, ]`.

Value

The provided solutions data frame reduced to just one row per meta cluster defined by the split vector.

Examples

```
# dl <- data_list(
#     list(subc_v, "subcortical_volume", "neuroimaging", "continuous"),
#     list(income, "household_income", "demographics", "continuous"),
#     list(pubertal, "pubertal_status", "demographics", "continuous"),
#     list(anxiety, "anxiety", "behaviour", "ordinal"),
#     list(depress, "depressed", "behaviour", "ordinal"),
#     uid = "unique_id"
# )
#
# sc <- snf_config(
#     dl = dl,
#     n_solutions = 20,
#     min_k = 20,
#     max_k = 50
# )
#
# sol_df <- batch_snf(dl, sc)
#
# ext_sol_df <- extend_solutions(
#     sol_df,
#     dl = dl,
#     min_pval = 1e-10 # p-values below 1e-10 will be thresholded to 1e-10
# )
#
# # Calculate pairwise similarities between cluster solutions
# sol_aris <- calc_aris(sol_df)
#
# # Extract hierarchical clustering order of the cluster solutions
# meta_cluster_order <- get_matrix_order(sol_aris)
#
# # Identify meta cluster boundaries with shiny app or trial and error
# # ari_hm <- meta_cluster_heatmap(sol_aris, order = meta_cluster_order)
# # shiny_annotator(ari_hm)
#
# # Result of meta cluster examination
# split_vec <- c(2, 5, 12, 17)
#
# ext_sol_df <- label_meta_clusters(ext_sol_df, split_vec, meta_cluster_order)
#
# # Extracting representative solutions from each defined meta cluster
# rep_solutions <- get_representative_solutions(sol_aris, ext_sol_df)
```

`income` *Mock ABCD income data*

Description

Like the mock data frame "abcd_h_income", but with "unique_id" as the "uid".

Like the mock data frame "abcd_cort_sa", but with "unique_id" as the "uid".

Usage

```
income
```

```
income
```

Format

`income`**:**

A data frame with 300 rows and 2 columns:

unique_id The unique identifier of the ABCD dataset

household_income Household income in 3 category levels (low = 1, medium = 2, high = 3)

`income`**:**

A data frame with 300 rows and 2 columns:

unique_id The unique identifier of the ABCD dataset

household_income Household income in 3 category levels (low = 1, medium = 2, high = 3)

Source

Though this data is no longer "real" ABCD data, the reference for using ABCD as a data source is below:

Data used in the preparation of this article were obtained from the Adolescent Brain Cognitive DevelopmentSM (ABCD) Study (https://abcdstudy.org), held in the NIMH Data Archive (NDA). This is a multisite, longitudinal study designed to recruit more than 10,000 children age 9-10 and follow them over 10 years into early adulthood. The ABCD Study® is supported by the National Institutes of Health and additional federal partners under award numbers U01DA041048, U01DA050989, U01DA051016, U01DA041022, U01DA051018, U01DA051037, U01DA050987, U01DA041174, U01DA041106, U01DA041117, U01DA041028, U01DA041134, U01DA050988, U01DA051039, U01DA041156, U01DA041025, U01DA041120, U01DA051038, U01DA041148, U01DA041093, U01DA041089, U24DA041123, U24DA041147. A full list of supporters is available at https://abcdstudy.org/federal-partners.html. A listing of participating sites and a complete listing of the study investigators can be found at https://abcdstudy.org/consortium_members/. ABCD consortium investigators designed and implemented the study and/or provided data but did not necessarily participate in the analysis or writing of this report. This manuscript reflects the views of the authors and may not reflect the opinions or views of the NIH or ABCD consortium investigators.

Though this data is no longer "real" ABCD data, the reference for using ABCD as a data source is below:

Data used in the preparation of this article were obtained from the Adolescent Brain Cognitive DevelopmentSM (ABCD) Study (https://abcdstudy.org), held in the NIMH Data Archive (NDA). This is a multisite, longitudinal study designed to recruit more than 10,000 children age 9-10 and follow them over 10 years into early adulthood. The ABCD Study® is supported by the National Institutes of Health and additional federal partners under award numbers U01DA041048, U01DA050989, U01DA051016, U01DA041022, U01DA051018, U01DA051037, U01DA050987, U01DA041174, U01DA041106, U01DA041117, U01DA041028, U01DA041134, U01DA050988, U01DA051039, U01DA041156, U01DA041025, U01DA041120, U01DA051038, U01DA041148, U01DA041093, U01DA041089, U24DA041123, U24DA041147. A full list of supporters is available at https://abcdstudy.org/federal-partners.html. A listing of participating sites and a complete listing of the study investigators can be found at https://abcdstudy.org/consortium_members/. ABCD consortium investigators designed and implemented the study and/or provided data but did not necessarily participate in the analysis or writing of this report. This manuscript reflects the views of the authors and may not reflect the opinions or views of the NIH or ABCD consortium investigators.

`is_data_list` *Test if the object is a data list*

Description

Given an object, returns `TRUE` if that object inherits from the `data_list` class.

Usage

```
is_data_list(x)
```

Arguments

`x`	An object.

Value

`TRUE` if the object inherits from the `data_list` class.

`jitter_plot` *Jitter plot separating a feature by cluster*

Description

Jitter plot separating a feature by cluster

Usage

```
jitter_plot(df, feature)
```

Arguments

`df`	A data.frame containing cluster column and the feature to plot.
`feature`	The feature to plot.

Value

A jitter+violin plot (class "gg", "ggplot") showing the distribution of a feature across clusters.

`label_meta_clusters` *Assign meta cluster labels to rows of a solutions data frame or extended solutions data frame*

Description

Given a solutions data frame or extended solutions data frame class object and a numeric vector indicating which rows correspond to which meta clusters, assigns meta clustering information to the "meta_clusters" attribute of the data frame.

Usage

```
label_meta_clusters(sol_df, split_vector, order = NULL)
```

Arguments

`sol_df`	A solutions data frame or extended solutions data frame to assign meta clusters to.
`split_vector`	A numeric vector indicating which rows of sol_df should be the split points for meta cluster labeling.
`order`	An optional numeric vector indicating how the solutions data frame should be reordered prior to meta cluster labeling. This vector can be obtained by running `get_matrix_order()` on an ARI matrix, which itself can be obtained by calling `calc_aris()` on a solutions data frame.

Value

A solutions data frame with a populated "meta_clusters" attribute.

Examples

```
#dl <- data_list(
#    list(cort_sa, "cortical_surface_area", "neuroimaging", "continuous"),
#    list(subc_v, "subcortical_volume", "neuroimaging", "continuous"),
#    list(income, "household_income", "demographics", "continuous"),
#    list(pubertal, "pubertal_status", "demographics", "continuous"),
#    uid = "unique_id"
#)
#
#set.seed(42)
#my_sc <- snf_config(
#    dl = dl,
#    n_solutions = 20,
#    min_k = 20,
#    max_k = 50
#)
#
#sol_df <- batch_snf(dl, my_sc)
#
#sol_df
#
#sol_aris <- calc_aris(sol_df)
#
#meta_cluster_order <- get_matrix_order(sol_aris)
#
## `split_vec` found by iteratively plotting ari_hm or by ?shiny_annotator()
#split_vec <- c(6, 10, 16)
#ari_hm <- meta_cluster_heatmap(
#    sol_aris,
#    order = meta_cluster_order,
#    split_vector = split_vec
#)
#
#mc_sol_df <- label_meta_clusters(
#    sol_df,
#    order = meta_cluster_order,
#    split_vector = split_vec
#)
#
#mc_sol_df
```

`label_propagate` *Label propagate cluster solutions to unclustered observations*

Description

Given a solutions data frame containing clustered observations and a data list containing those clustered observations as well as additional to-be-clustered observations, this function will re-run SNF to generate a similarity matrix of all observations and use the label propagation algorithm to assigned predicted clusters to the unclustered observations.

Usage

```
label_propagate(partial_sol_df, full_dl, verbose = FALSE)
```

Arguments

`partial_sol_df`	A solutions data frame derived from the training set.
`full_dl`	A data list containing observations from both the training and testing sets.
`verbose`	If TRUE, output progress to console.

Value

A data frame with one row per observation containing a column for UIDs, a column for whether the observation was in the train (original) or test (held out) set, and one column per row of the solutions data frame indicating the original and propagated clusters.

Examples

```
## Function to identify obervations with complete data
#uids_with_complete_obs <- get_complete_uids(
#    list(subc_v, income, pubertal, anxiety, depress),
#    uid = "unique_id"
#)
#
## Dataframe assigning 80% of observations to train and 20% to test
#train_test_split <- train_test_assign(
#    train_frac = 0.8,
#    uids = uids_with_complete_obs
#)
#
## Pulling the training and testing observations specifically
#train_obs <- train_test_split$"train"
#test_obs <- train_test_split$"test"
#
## Partition a training set
#train_subc_v <- subc_v[subc_v$"unique_id" %in% train_obs, ]
#train_income <- income[income$"unique_id" %in% train_obs, ]
#train_pubertal <- pubertal[pubertal$"unique_id" %in% train_obs, ]
#train_anxiety <- anxiety[anxiety$"unique_id" %in% train_obs, ]
#train_depress <- depress[depress$"unique_id" %in% train_obs, ]
#
## Partition a test set
#test_subc_v <- subc_v[subc_v$"unique_id" %in% test_obs, ]
#test_income <- income[income$"unique_id" %in% test_obs, ]
```

```
#test_pubertal <- pubertal[pubertal$"unique_id" %in% test_obs, ]
#test_anxiety <- anxiety[anxiety$"unique_id" %in% test_obs, ]
#test_depress <- depress[depress$"unique_id" %in% test_obs, ]
#
## Find cluster solutions in the training set
#train_dl <- data_list(
#    list(train_subc_v, "subc_v", "neuroimaging", "continuous"),
#    list(train_income, "household_income", "demographics", "continuous"),
#    list(train_pubertal, "pubertal_status", "demographics", "continuous"),
#    uid = "unique_id"
#)
#
## We'll pick a solution that has good separation over our target features
#train_target_dl <- data_list(
#    list(train_anxiety, "anxiety", "behaviour", "ordinal"),
#    list(train_depress, "depressed", "behaviour", "ordinal"),
#    uid = "unique_id"
#)
#
#sc <- snf_config(
#    train_dl,
#    n_solutions = 5,
#    min_k = 10,
#    max_k = 30
#)
#
#train_sol_df <- batch_snf(
#    train_dl,
#    sc,
#    return_sim_mats = TRUE
#)
#
#ext_sol_df <- extend_solutions(
#    train_sol_df,
#    train_target_dl
#)
#
## Determining solution with the lowest minimum p-value
#lowest_min_pval <- min(ext_sol_df$"min_pval")
#which(ext_sol_df$"min_pval" == lowest_min_pval)
#top_row <- ext_sol_df[1, ]
#
## Propagate that solution to the observations in the test set
## data list below has both training and testing observations
#full_dl <- data_list(
#    list(subc_v, "subc_v", "neuroimaging", "continuous"),
#    list(income, "household_income", "demographics", "continuous"),
#    list(pubertal, "pubertal_status", "demographics", "continuous"),
#    uid = "unique_id"
#)
#
## Use the solutions data frame from the training observations and the data list
## from the training and testing observations to propagate labels to the test observations
```

```
#propagated_labels <- label_propagate(top_row, full_dl)
#
#propagated_labels_all <- label_propagate(ext_sol_df, full_dl)
#
#head(propagated_labels_all)
#tail(propagated_labels_all)
```

linear_adjust *Linearly correct data list by features with unwanted signal*

Description

Given a data list to correct and another data list of categorical features to linearly adjust for, corrects the first data list based on the residuals of the linear model relating the numeric features in the first data list to the unwanted signal features in the second data list.

Usage

```
linear_adjust(dl, unwanted_signal_list, sig_digs = NULL)
```

Arguments

dl	A nested list of input data from data_list().
unwanted_signal_list	A data list of categorical features that should have their mean differences removed in the first data list.
sig_digs	Number of significant digits to round the residuals to.

Value

A data list ("list") in which each data component has been converted to contain residuals off of the linear model built against the features in the unwanted_signal_list.

Examples

```
has_tutor <- sample(c(1, 0), size = 9, replace = TRUE)
math_score <- 70 + 30 * has_tutor + rnorm(9, mean = 0, sd = 5)

math_df <- data.frame(uid = paste0("id_", 1:9), math = math_score)
tutor_df <- data.frame(uid = paste0("id_", 1:9), tutor = has_tutor)

dl <- data_list(
    list(math_df, "math_score", "school", "continuous"),
    uid = "uid"
)

adjustment_dl <- data_list(
    list(tutor_df, "tutoring", "school", "categorical"),
    uid = "uid"
)

adjusted_dl <- linear_adjust(dl, adjustment_dl)

adjusted_dl[[1]]$"data"$"math"

# Equivalent to:
as.numeric(resid(lm(math_score ~ has_tutor)))
```

`mc_manhattan_plot` *Manhattan plot of feature-meta cluster associaiton p-values*

Description

Given a data frame of representative meta cluster solutions (see `get_representative_solutions()`, returns a Manhattan plot for showing feature separation across all features in provided data/target lists.

Usage

```
mc_manhattan_plot(
  ext_sol_df,
  dl = NULL,
  target_dl = NULL,
  variable_order = NULL,
  neg_log_pval_thresh = 5,
  threshold = NULL,
  point_size = 5,
  text_size = 20,
  plot_title = NULL,
  xints = NULL,
  hide_x_labels = FALSE,
  domain_colours = NULL
)
```

Arguments

`ext_sol_df` A sol_df that contains "_pval" columns containing the values to be plotted. This object is the output of `extend_solutions()`.

`dl` List of data frames containing data information.

`target_dl` List of data frames containing target information.

`variable_order` Order of features to be displayed in the plot.

`neg_log_pval_thresh`
Threshold for negative log p-values.

`threshold` p-value threshold to plot horizontal dashed line at.

`point_size` Size of points in the plot.

text_size	Size of text in the plot.
plot_title	Title of the plot.
xints	Either "outcomes" or a vector of numeric values to plot vertical lines at.
hide_x_labels	If TRUE, hides x-axis labels.
domain_colours	Named vector of colours for domains.

Value

A Manhattan plot (class "gg", "ggplot") showing the association p-values of features against each solution in the provided solutions data frame, stratified by meta cluster label.

Examples

```
# dl <- data_list(
#     list(subc_v, "subcortical_volume", "neuroimaging", "continuous"),
#     list(income, "household_income", "demographics", "continuous"),
#     list(pubertal, "pubertal_status", "demographics", "continuous"),
#     list(anxiety, "anxiety", "behaviour", "ordinal"),
#     list(depress, "depressed", "behaviour", "ordinal"),
#     uid = "unique_id"
# )
#
# sc <- snf_config(
#     dl = dl,
#     n_solutions = 20,
#     min_k = 20,
#     max_k = 50
# )
#
# sol_df <- batch_snf(dl, sc)
#
# ext_sol_df <- extend_solutions(
#     sol_df,
#     dl = dl,
#     min_pval = 1e-10 # p-values below 1e-10 will be thresholded to 1e-10
# )
#
# # Calculate pairwise similarities between cluster solutions
# sol_aris <- calc_aris(sol_df)
#
# # Extract hierarchical clustering order of the cluster solutions
# meta_cluster_order <- get_matrix_order(sol_aris)
#
# # Identify meta cluster boundaries with shiny app or trial and error
# # ari_hm <- meta_cluster_heatmap(sol_aris, order = meta_cluster_order)
# # shiny_annotator(ari_hm)
#
# # Result of meta cluster examination
# split_vec <- c(2, 5, 12, 17)
#
# ext_sol_df <- label_meta_clusters(ext_sol_df, split_vec, meta_cluster_order)
```

```
#
# # Extracting representative solutions from each defined meta cluster
# rep_solutions <- get_representative_solutions(sol_aris, ext_sol_df)
#
# mc_manhattan <- mc_manhattan_plot(
#     rep_solutions,
#     dl = dl,
#     point_size = 3,
#     text_size = 12,
#     plot_title = "Feature-Meta Cluster Associations",
#     threshold = 0.05,
#     neg_log_pval_thresh = 5
# )
#
# mc_manhattan
```

merge.data_list *Merge observations between two compatible data lists*

Description

Join two data lists with the same components (data frames) but separate observations. To instead merge two data lists that have the same observations but different components, simply use `c()`.

Usage

```
## S3 method for class 'data_list'
merge(x, y, ...)
```

Arguments

`x`	The first data list to merge.
`y`	The second data list to merge.
`...`	Additional arguments passed into merge function.

Value

A data list ("list"-class object) containing the observations of both provided data lists.

`merge.snf_config` *Merge method for SNF config objects*

Description

Merge method for SNF config objects

Usage

```
## S3 method for class 'snf_config'
merge(x, y, reset_indices = TRUE, ...)
```

Arguments

`x`	SNF config to merge.
`y`	SNF config to merge.
`reset_indices`	If TRUE (default), re-labels the "solutions" indices in the config from 1 to the number of defined settings.
`...`	Additional arguments passed into merge function.

Value

An SNF config combining the rows of both prior configurations.

`merge_df_list` *Merge list of data frames into a single data frame*

Description

This helper function combines all data frames in a single-level list into a single data frame.

Usage

```
merge_df_list(df_list, join = "inner", uid = "uid", no_na = FALSE)
```

Arguments

`df_list`	list of data frames.
`join`	String indicating if join should be "inner" or "full".
`uid`	Column name to join on. Default is "uid".
`no_na`	Whether to remove NA values from the merged data frame.

Value

Inner join of all data frames in list.

Examples

```
merge_df_list(list(income, pubertal), uid = "unique_id")
```

meta_cluster_heatmap *Heatmap of pairwise adjusted rand indices between solutions*

Description

Heatmap of pairwise adjusted rand indices between solutions

Usage

```
meta_cluster_heatmap(
  aris,
  order = NULL,
  cluster_rows = FALSE,
  cluster_columns = FALSE,
  log_graph = FALSE,
  scale_diag = "none",
  min_colour = "#282828",
  max_colour = "firebrick2",
 col = circlize::colorRamp2(c(min(aris), max(aris)), c(min_colour, max_colour)),
  ...
)
```

Arguments

`aris`	Matrix of adjusted rand indices from `calc_aris()`
`order`	Numeric vector containing row order of the heatmap.
`cluster_rows`	Whether rows should be clustered.
`cluster_columns`	Whether columns should be clustered.
`log_graph`	If TRUE, log transforms the graph.
`scale_diag`	Method of rescaling matrix diagonals. Can be "none" (don't change diagonals), "mean" (replace diagonals with average value of off-diagonals), or "zero" (replace diagonals with 0).
`min_colour`	Colour used for the lowest value in the heatmap.
`max_colour`	Colour used for the highest value in the heatmap.
`col`	Colour ramp to use for the heatmap.
`...`	Additional parameters passed to `similarity_matrix_heatmap()`, the function that this function wraps.

Value

Returns a heatmap (class "Heatmap" from package ComplexHeatmap) that displays the pairwise adjusted Rand indices (similarities) between the cluster solutions of the provided solutions data frame.

Examples

```
#dl <- data_list(
#    list(cort_sa, "cortical_surface_area", "neuroimaging", "continuous"),
#    list(subc_v, "subcortical_volume", "neuroimaging", "continuous"),
#    list(income, "household_income", "demographics", "continuous"),
#    list(pubertal, "pubertal_status", "demographics", "continuous"),
#    uid = "unique_id"
#)
#
#set.seed(42)
#my_sc <- snf_config(
#    dl = dl,
#    n_solutions = 20,
#    min_k = 20,
#    max_k = 50
#)
#
#sol_df <- batch_snf(dl, my_sc)
#
#sol_df
#
#sol_aris <- calc_aris(sol_df)
#
#meta_cluster_order <- get_matrix_order(sol_aris)
#
## `split_vec` found by iteratively plotting ari_hm or by ?shiny_annotator()
#split_vec <- c(6, 10, 16)
#ari_hm <- meta_cluster_heatmap(
#    sol_aris,
#    order = meta_cluster_order,
#    split_vector = split_vec
#)
```

methylation_df *Modification of SNFtool mock data frame "Data2"*

Description

Modification of SNFtool mock data frame "Data2"

Usage

```
methylation_df
```

Format

`methylation_df`**:**

A data frame with 200 rows and 3 columns:

gene_1_expression Mock gene methylation feature

gene_2_expression Mock gene methylation feature

patient_id Random three-digit number uniquely identifying the patient

Source

This data came from the SNFtool package, with slight modifications.

`new_solutions_df` *Constructor for* `solutions_df` *class object*

Description

Constructor for `solutions_df` class object

Usage

```
new_solutions_df(sol_dfl)
```

Arguments

`sol_dfl`	A solutions data frame-like object to be validated and converted into a solutions data frame.

Value

A `solutions_df` class object.

`n_features` *Extract number of features stored in an object*

Description

Extract number of features stored in an object

Usage

```
n_features(x)
```

Arguments

`x`	The object to extract number of features from.

Value

The number of features in x.

n_observations *Extract number of observations stored in an object*

Description

Extract number of observations stored in an object

Usage

```
n_observations(x)
```

Arguments

x	The object to extract number of observations from.

Value

The number of observations in x.

print.ari_matrix *Print method for class* ari_matrix

Description

Custom formatted print for weights matrices that outputs information about feature weights functions to the console.

Usage

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

Arguments

x	A ari_matrix class object.
...	Other arguments passed to print (not used in this function)

Value

Function prints to console but does not return any value.

`print.clust_fns_list` *Print method for class* `clust_fns_list`

Description

Custom formatted print for clustering functions list objects that outputs information about the contained clustering functions to the console.

Usage

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

Arguments

`x`	A `clust_fns_list` class object.
`...`	Other arguments passed to `print` (not used in this function)

Value

Function prints to console but does not return any value.

`print.data_list` *Print method for class* `data_list`

Description

Custom formatted print for data list objects that outputs information about the contained observations and components to the console.

Usage

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

Arguments

`x`	A `data_list` class object.
`...`	Other arguments passed to `print` (not used in this function)

Value

Function prints to console but does not return any value.

print.dist_fns_list *Print method for class* dist_fns_list

Description

Custom formatted print for distance metrics list objects that outputs information about the contained distance metrics to the console.

Usage

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

Arguments

x	A dist_fns_list class object.
...	Other arguments passed to print (not used in this function)

Value

Function prints to console but does not return any value.

print.ext_solutions_df *Print method for class* ext_solutions_df

Description

Custom formatted print for extended solutions data frame class objects.

Usage

```
## S3 method for class 'ext_solutions_df'
print(x, n = NULL, ...)
```

Arguments

x	A ext_solutions_df class object.
n	Number of rows to print, passed into tibble::print.tbl_df().
...	Other arguments passed to print (not used in this function).

Value

Function prints to console but does not return any value.

`print.settings_df` *Print method for class* `settings_df`

Description

Custom formatted print for settings data frame that outputs information about SNF hyperparameters to the console.

Usage

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

Arguments

`x`	A `settings_df` class object.
`...`	Other arguments passed to `print` (not used in this function)

Value

Function prints to console but does not return any value.

`print.snf_config` *Print method for class* `snf_config`

Description

Custom formatted print for SNF config

Usage

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

Arguments

`x`	A `snf_config` class object.
`...`	Other arguments passed to `print` (not used in this function)

Value

Function prints to console but does not return any value.

print.solutions_df *Print method for class* solutions_df

Description

Custom formatted print for weights matrices that outputs information about feature weights functions to the console.

Usage

```
## S3 method for class 'solutions_df'
print(x, n = NULL, tips = TRUE, ...)
```

Arguments

x	A weights_matrix class object.
n	Number of rows to print, passed into tibble::print.tbl_df().
tips	If TRUE, include lines on how to print more rows / transposed.
...	Other arguments passed to print (not used in this function).

Value

Function prints to console but does not return any value.

print.t_ext_solutions_df *Print method for class* t_ext_solutions_df

Description

Custom formatted print for transposed solutions data frame class objects.

Usage

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

Arguments

x	A t_solutions_df class object.
...	Other arguments passed to print (not used in this function)

Value

Function prints to console but does not return any value.

print.t_solutions_df *Print method for class* t_solutions_df

Description

Custom formatted print for transposed solutions data frame class objects.

Usage

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

Arguments

x	A t_solutions_df class object.
...	Other arguments passed to print (not used in this function)

Value

Function prints to console but does not return any value.

print.weights_matrix *Print method for class* weights_matrix

Description

Custom formatted print for weights matrices that outputs information about feature weights functions to the console.

Usage

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

Arguments

x	A weights_matrix class object.
...	Other arguments passed to print (not used in this function)

Value

Function prints to console but does not return any value.

`pubertal` *Mock ABCD pubertal status data*

Description

Like the mock data frame "abcd_pubertal", but with "unique_id" as the "uid".

Usage

```
pubertal
```

Format

`pubertal`**:**

A data frame with 275 rows and 2 columns:

unique_id The unique identifier of the ABCD dataset

pubertal_status Average reported pubertal status between child and parent (1-5 categorical scale)

Source

Though this data is no longer "real" ABCD data, the reference for using ABCD as a data source is below:

Data used in the preparation of this article were obtained from the Adolescent Brain Cognitive DevelopmentSM (ABCD) Study (https://abcdstudy.org), held in the NIMH Data Archive (NDA). This is a multisite, longitudinal study designed to recruit more than 10,000 children age 9-10 and follow them over 10 years into early adulthood. The ABCD Study® is supported by the National Institutes of Health and additional federal partners under award numbers U01DA041048, U01DA050989, U01DA051016, U01DA041022, U01DA051018, U01DA051037, U01DA050987, U01DA041174, U01DA041106, U01DA041117, U01DA041028, U01DA041134, U01DA050988, U01DA051039, U01DA041156, U01DA041025, U01DA041120, U01DA051038, U01DA041148, U01DA041093, U01DA041089, U24DA041123, U24DA041147. A full list of supporters is available at https://abcdstudy.org/federal-partners.html. A listing of participating sites and a complete listing of the study investigators can be found at https://abcdstudy.org/consortium_members/. ABCD consortium investigators designed and implemented the study and/or provided data but did not necessarily participate in the analysis or writing of this report. This manuscript reflects the views of the authors and may not reflect the opinions or views of the NIH or ABCD consortium investigators.

`pval_heatmap` *Heatmap of p-values*

Description

Heatmap of p-values

Usage

```
pval_heatmap(
  ext_sol_df,
  order = NULL,
  cluster_columns = TRUE,
  cluster_rows = FALSE,
  show_row_names = FALSE,
  show_column_names = TRUE,
  min_colour = "red2",
  max_colour = "white",
  legend_breaks = c(0, 1),
  col = circlize::colorRamp2(legend_breaks, c(min_colour, max_colour)),
 heatmap_legend_param = list(color_bar = "continuous", title = "p-value", at = c(0, 1)),
  rect_gp = grid::gpar(col = "black"),
  column_split_vector = NULL,
  row_split_vector = NULL,
  column_split = NULL,
  row_split = NULL,
  ...
)
```

Arguments

`ext_sol_df`	An ext_solutions_df class object (produced from the function `extend_solutions`.
`order`	Numeric vector containing row order of the heatmap.
`cluster_columns`	Whether columns should be sorted by hierarchical clustering.
`cluster_rows`	Whether rows should be sorted by hierarchical clustering.
`show_row_names`	Whether row names should be shown.
`show_column_names`	Whether column names should be shown.
`min_colour`	Colour used for the lowest value in the heatmap.
`max_colour`	Colour used for the highest value in the heatmap.
`legend_breaks`	Numeric vector of breaks for the legend.
`col`	Colour function for `ComplexHeatmap::Heatmap()`
`heatmap_legend_param`	Legend function for `ComplexHeatmap::Heatmap()`
`rect_gp`	Cell border function for `ComplexHeatmap::Heatmap()`
`column_split_vector`	Vector of indices to split columns by.
`row_split_vector`	Vector of indices to split rows by.
`column_split`	Standard parameter of `ComplexHeatmap::Heatmap`.
`row_split`	Standard parameter of `ComplexHeatmap::Heatmap`.
`...`	Additional parameters passed to `ComplexHeatmap::Heatmap`.

Value

Returns a heatmap (class "Heatmap" from package ComplexHeatmap) that displays the provided p-values.

Examples

```
#dl <- data_list(
#    list(income, "household_income", "demographics", "ordinal"),
#    list(pubertal, "pubertal_status", "demographics", "continuous"),
#    list(fav_colour, "favourite_colour", "demographics", "categorical"),
#    list(anxiety, "anxiety", "behaviour", "ordinal"),
#    list(depress, "depressed", "behaviour", "ordinal"),
#    uid = "unique_id"
#)
#
#sc <- snf_config(
#    dl,
#    n_solutions = 4,
#    dropout_dist = "uniform",
#    max_k = 50
#)
#
#sol_df <- batch_snf(dl, sc)
#
#ext_sol_df <- extend_solutions(sol_df, dl)
#
#pval_heatmap(ext_sol_df)
```

---

quality_measures *Quality metrics*

Description

These functions calculate conventional metrics of cluster solution quality.

Usage

```
calculate_silhouettes(sol_df)

calculate_dunn_indices(sol_df)

calculate_db_indices(sol_df)
```

Arguments

sol_df	A `solutions_df` class object created by `batch_snf()` with the parameter `return_sim_mats = TRUE`.

Details

calculate_silhouettes: A wrapper for `cluster::silhouette` that calculates silhouette scores for all cluster solutions in a provided solutions data frame. Silhouette values range from -1 to +1 and indicate an overall ratio of how close together observations within a cluster are to how far apart observations across clusters are. You can learn more about interpreting the results of this function by calling `?cluster::silhouette`.

calculate_dunn_indices: A wrapper for `clv::clv.Dunn` that calculates Dunn indices for all cluster solutions in a provided solutions data frame. Dunn indices, like silhouette scores, similarly reflect similarity within clusters and separation across clusters. You can learn more about interpreting the results of this function by calling `?clv::clv.Dunn`.

calculate_db_indices: A wrapper for `clv::clv.Davies.Bouldin` that calculates Davies-Bouldin indices for all cluster solutions in a provided solutions data frame. These values can be interpreted similarly as those above. You can learn more about interpreting the results of this function by calling `?clv::clv.Davies.Bouldin`.

Value

A list of `silhouette` class objects, a vector of Dunn indices, or a vector of Davies-Bouldin indices depending on which function was used.

Examples

```
input_dl <- data_list(
    list(gender_df, "gender", "demographics", "categorical"),
    list(diagnosis_df, "diagnosis", "clinical", "categorical"),
    uid = "patient_id"
)

sc <- snf_config(input_dl, n_solutions = 5)

sol_df <- batch_snf(input_dl, sc, return_sim_mats = TRUE)

# calculate Davies-Bouldin indices
davies_bouldin_indices <- calculate_db_indices(sol_df)

# calculate Dunn indices
dunn_indices <- calculate_dunn_indices(sol_df)

# calculate silhouette scores
silhouette_scores <- calculate_silhouettes(sol_df)
```

`random_removal` *Generate random removal sequence*

Description

Helper function to contribute to rows within the settings data frame. Number of columns removed follows a uniform or exponential probability distribution.

Usage

```
random_removal(
  columns,
  min_removed_inputs,
  max_removed_inputs,
  dropout_dist = "exponential"
)
```

Arguments

`columns`	Columns of the settings_df that are passed in
`min_removed_inputs`	The smallest number of input data frames that may be randomly removed.
`max_removed_inputs`	The largest number of input data frames that may be randomly removed.
`dropout_dist`	Indication of how input data frames should be dropped. can be "none" (no dropout), "uniform" (uniformly draw number between min and max removed inputs), or "exponential" (like uniform, but using an exponential distribution; default).

Value

inclusions_df data frame that can be rbind'ed to the settings_df

---

rbind.ext_solutions_df *Row-binding of solutions data frame class objects.*

Description

Row-binding of solutions data frame class objects.

Usage

```
## S3 method for class 'ext_solutions_df'
rbind(..., reset_indices = FALSE)
```

Arguments

`...`	An arbitrary number of `ext_solutions_df` class objects.
`reset_indices`	If TRUE, re-labels the "solutions" indices in the solutions data frame from 1 to the number of defined settings.

Value

An `ext_solutions_df` class object.

rbind.solutions_df *Row-binding of solutions data frame class objects.*

Description

Row-binding of solutions data frame class objects.

Usage

```
## S3 method for class 'solutions_df'
rbind(..., reset_indices = FALSE)
```

Arguments

`...`	An arbitrary number of `solutions_df` class objects.
`reset_indices`	If TRUE, re-labels the "solutions" indices in the solutions data frame from 1 to the number of defined settings.

Value

A `solutions_df` class object.

rename_dl *Rename features in a data list*

Description

Rename features in a data list

Usage

```
rename_dl(dl, name_mapping)
```

Arguments

`dl`	A nested list of input data from `data_list()`.
`name_mapping`	A named vector where the values are the features to be renamed and the names are the new names for those features.

Value

A data list ("list"-class object) with adjusted feature names.

Examples

```
library(metasnf)

dl <- data_list(
    list(pubertal, "pubertal_status", "demographics", "continuous"),
    list(anxiety, "anxiety", "behaviour", "ordinal"),
    list(depress, "depressed", "behaviour", "ordinal"),
    uid = "unique_id"
)

summary(dl, "feature")

name_changes <- c(
    "anxiety_score" = "cbcl_anxiety_r",
    "depression_score" = "cbcl_depress_r"
)

dl <- rename_dl(dl, name_changes)

summary(dl, "feature")
```

resample *Helper resample function found in ?sample*

Description

Like sample, but when given a single value x, returns back that single value instead of a random value from 1 to x.

Usage

```
resample(x, ...)
```

Arguments

`x`	Vector or single value to sample from
`...`	Remaining arguments for base::sample function

Value

Numeric vector result of running base::sample.

save_heatmap *Save a heatmap object to a file*

Description

Save a heatmap object to a file

Usage

```
save_heatmap(heatmap, path, width = 480, height = 480, res = 100)
```

Arguments

heatmap	The heatmap object to save.
path	The path to save the heatmap to.
width	The width of the heatmap.
height	The height of the heatmap.
res	The resolution of the heatmap.

Value

Does not return any value. Saves heatmap to file.

settings_df *Build a settings data frame*

Description

The settings_df is a data frame whose rows completely specify the hyperparameters and decisions required to transform individual input data frames (found in a data list, see ?data_list) into a single similarity matrix through SNF. The format of the settings data frame is as follows:

- A column named "solution": This column is used to keep track of the rows and should have integer values only.
- A column named "alpha": This column contains the value of the alpha hyperparameter that will be used on that run of the SNF pipeline.
- A column named "k": Like above, but for the K (nearest neighbours) hyperparameter.
- A column named "t": Like above, but for the t (number of iterations) hyperparameter.
- A column named "snf_scheme": Which of 3 pre-defined schemes will be used to integrate the data frames of the data list into a final fused network. The purpose of varying these schemes is primarily to increase the diversity of the generated cluster solutions.

  - A value of 1 corresponds to the "individual" scheme, in which all data frames are directly merged by SNF into the final fused network. This scheme corresponds to the approach shown in the original SNF paper.
  - A value of 2 corresponds to the "two-step" scheme, in which all data frames witin a domain are first merged into a domain-specific fused network. Next, domain-specific networks are fused once more by SNF into the final fused network. This scheme is useful for fairly re-weighting SNF pipelines with unequal numbers of data frames across domains.
  - A value of 3 corresponds to the "domain" scheme, in which all data frames within a domain are first concatenated into a single domain- specific data frame before being merged by SNF into the final fused network. This approach serves as an alternative way to reweight SNF pipelines with unequal numbers of data frames across domains. You can learn more about this parameter here: https://branchlab.github.io/metasnf/articles/snf_schemes.html.
- A column named "clust_alg": Specification of which clustering algorithm will be applied to the final similarity matrix. By default, this column can take on the integer values 1 or 2, which correspond to spectral clustering where the number of clusters is determined by the eigengap or rotation cost heuristic respectively. You can learn more about this parameter here: https://branchlab.github.io/metasnf/articles/clustering_algorithms.html.
- A column named "cnt_dist": Specification of which distance metric will be used for data frames of purely continuous data. You can learn about this metric and its defaults here: https://branchlab.github.io/metasnf/articles/distance_metrics.html
- A column named "dsc_dist": Like above, but for discrete data frames.
- A column named "ord_dist": Like above, but for ordinal data frames.
- A column named "cat_dist": Like above, but for categorical data frames.
- A column named "mix_dist": Like above, but for mixed-type (e.g., both categorical and discrete) data frames.
- One column for every input data frame in the corresponding data list which can either have the value of 0 or 1. The name of the column should be formatted as "inc_[]" where the square brackets are replaced with the name (as found in dl_summary(dl)$"name") of each data frame. When 0, that data frame will be excluded from that run of the SNF pipeline. When 1, that data frame will be included.

Usage

```
settings_df(
  dl,
  n_solutions = 0,
  min_removed_inputs = 0,
  max_removed_inputs = length(dl) - 1,
  dropout_dist = "exponential",
  min_alpha = NULL,
  max_alpha = NULL,
  min_k = NULL,
  max_k = NULL,
  min_t = NULL,
  max_t = NULL,
  alpha_values = NULL,
  k_values = NULL,
  t_values = NULL,
  possible_snf_schemes = c(1, 2, 3),
  clustering_algorithms = NULL,
  continuous_distances = NULL,
  discrete_distances = NULL,
  ordinal_distances = NULL,
  categorical_distances = NULL,
  mixed_distances = NULL,
  dfl = NULL,
  snf_input_weights = NULL,
  snf_domain_weights = NULL,
  retry_limit = 10,
  allow_duplicates = FALSE
)
```

Arguments

`dl` A nested list of input data from `data_list()`.

`n_solutions` Number of rows to generate for the settings data frame.

`min_removed_inputs` The smallest number of input data frames that may be randomly removed. By default, 0.

`max_removed_inputs` The largest number of input data frames that may be randomly removed. By default, this is 1 less than all the provided input data frames in the data list.

`dropout_dist` Parameter controlling how the random removal of input data frames should occur. Can be "none" (no input data frames are randomly removed), "uniform" (uniformly sample between min_removed_inputs and max_removed_inputs to determine number of input data frames to remove), or "exponential" (pick number of input data frames to remove by sampling from min_removed_inputs to max_removed_inputs with an exponential distribution; the default).

`min_alpha` The minimum value that the alpha hyperparameter can have. Random assigned value of alpha for each row will be obtained by uniformly sampling numbers between `min_alpha` and `max_alpha` at intervals of 0.1. Cannot be used in conjunction with the `alpha_values` parameter.

`max_alpha` The maximum value that the alpha hyperparameter can have. See `min_alpha` parameter. Cannot be used in conjunction with the `alpha_values` parameter.

`min_k` The minimum value that the k hyperparameter can have. Random assigned value of k for each row will be obtained by uniformly sampling numbers between `min_k` and `max_k` at intervals of 1. Cannot be used in conjunction with the `k_values` parameter.

`max_k` The maximum value that the k hyperparameter can have. See `min_k` parameter. Cannot be used in conjunction with the `k_values` parameter.

`min_t` The minimum value that the t hyperparameter can have. Random assigned value of t for each row will be obtained by uniformly sampling numbers between `min_t` and `max_t` at intervals of 1. Cannot be used in conjunction with the `t_values` parameter.

`max_t` The maximum value that the t hyperparameter can have. See `min_t` parameter. Cannot be used in conjunction with the `t_values` parameter.

`alpha_values` A number or numeric vector of a set of possible values that alpha can take on. Value will be obtained by uniformly sampling the vector. Cannot be used in conjunction with the `min_alpha` or `max_alpha` parameters.

`k_values` A number or numeric vector of a set of possible values that k can take on. Value will be obtained by uniformly sampling the vector. Cannot be used in conjunction with the `min_k` or `max_k` parameters.

`t_values` A number or numeric vector of a set of possible values that t can take on. Value will be obtained by uniformly sampling the vector. Cannot be used in conjunction with the `min_t` or `max_t` parameters.

`possible_snf_schemes`

A vector containing the possible snf_schemes to uniformly randomly select from. By default, the vector contains all 3 possible schemes: c(1, 2, 3). 1 corresponds to the "individual" scheme, 2 corresponds to the "domain" scheme, and 3 corresponds to the "twostep" scheme.

`clustering_algorithms`

A list of clustering algorithms to uniformly randomly pick from when clustering. When not specified, randomly select between spectral clustering using the eigen-gap heuristic and spectral clustering using the rotation cost heuristic. See ?clust_fns_list for more details on running custom clustering algorithms.

`continuous_distances`

A vector of continuous distance metrics to use when a custom dist_fns_list is provided.

`discrete_distances`

A vector of categorical distance metrics to use when a custom dist_fns_list is provided.

`ordinal_distances`

A vector of categorical distance metrics to use when a custom dist_fns_list is provided.

`categorical_distances`

A vector of categorical distance metrics to use when a custom dist_fns_list is provided.

`mixed_distances`

A vector of mixed distance metrics to use when a custom dist_fns_list is provided.

`dfl` List containing distance metrics to vary over. See ?generate_dist_fns_list.

`snf_input_weights`

Nested list containing weights for when SNF is used to merge individual input measures (see ?generate_snf_weights)

`snf_domain_weights`

Nested list containing weights for when SNF is used to merge domains (see ?generate_snf_weights) retry_limit The maximum number of attempts to generate a novel row. This function does not return matrices with identical rows. As the range of requested possible settings tightens and the number of requested rows increases, the risk of randomly generating a row that already exists increases. If a new random row has matched an existing row retry_limit number of times, the function will terminate.

allow_duplicates If TRUE, enables creation of a settings data frame with duplicate non-feature weighting related hyperparameters. This function should only be used when paired with a custom weights matrix that has non-duplicate rows.

Value

A settings data frame

---

shiny_annotator *Launch a shiny app to identify meta cluster boundaries*

Description

This function calls the htShiny() function from the package InteractiveComplexHeatmap to assist users in identifying the indices of the boundaries between meta clusters in a meta cluster heatmap. By providing a heatmap of inter-solution similarities (obtained through meta_cluster_heatmap()), users can click on positions within the heatmap that appear to meaningfully separate major sets of similar cluster solutions by visual inspection. The corresponding indices of the clicked positions are printed to the console and also shown within the app. This function can only run from an interactive session of R.

Usage

```
shiny_annotator(ari_heatmap)
```

Arguments

ari_heatmap Heatmap of ARIs to divide into meta clusters.

Value

Does not return any value. Launches interactive shiny applet.

Examples

```
#dl <- data_list(
#    list(cort_sa, "cortical_surface_area", "neuroimaging", "continuous"),
#    list(subc_v, "subcortical_volume", "neuroimaging", "continuous"),
#    list(income, "household_income", "demographics", "continuous"),
#    list(pubertal, "pubertal_status", "demographics", "continuous"),
#    uid = "unique_id"
#)
```

```
#
#set.seed(42)
#my_sc <- snf_config(
#    dl = dl,
#    n_solutions = 20,
#    min_k = 20,
#    max_k = 50
#)
#
#sol_df <- batch_snf(dl, my_sc)
#
#sol_aris <- calc_aris(sol_df)
#
#meta_cluster_order <- get_matrix_order(sol_aris)
#
#ari_hm <- meta_cluster_heatmap(sol_aris, order = meta_cluster_order)
#
## Click on meta cluster boundaries to obtain `split_vec` values
#shiny_annotator(ari_hm)
#
#split_vec <- c(6, 10, 16)
#
#ari_hm <- meta_cluster_heatmap(
#    sol_aris,
#    order = meta_cluster_order,
#    split_vector = split_vec
#)
```

---

similarity_matrix_heatmap

Plot heatmap of similarity matrix

Description

Plot heatmap of similarity matrix

Usage

```
similarity_matrix_heatmap(
  similarity_matrix,
  order = NULL,
  cluster_solution = NULL,
  scale_diag = "mean",
  log_graph = TRUE,
  cluster_rows = FALSE,
  cluster_columns = FALSE,
  show_row_names = FALSE,
  show_column_names = FALSE,
  data = NULL,
  left_bar = NULL,
  right_bar = NULL,
  top_bar = NULL,
  bottom_bar = NULL,
  left_hm = NULL,
  right_hm = NULL,
  top_hm = NULL,
  bottom_hm = NULL,
  annotation_colours = NULL,
  min_colour = NULL,
  max_colour = NULL,
  split_vector = NULL,
  row_split = NULL,
  column_split = NULL,
  ...
)
```

Arguments

`similarity_matrix`
A similarity matrix

`order` Vector of numbers to reorder the similarity matrix (and data if provided). Overwrites ordering specified by cluster_solution param.

`cluster_solution`
Row of a solutions data frame or column of a transposed solutions data frame.

`scale_diag` Method of rescaling matrix diagonals. Can be "none" (don't change diagonals), "mean" (replace diagonals with average value of off-diagonals), or "zero" (replace diagonals with 0).

`log_graph` If TRUE, log transforms the graph.

`cluster_rows` Parameter for ComplexHeatmap::Heatmap.

`cluster_columns`
Parameter for ComplexHeatmap::Heatmap.

`show_row_names` Parameter for ComplexHeatmap::Heatmap.

`show_column_names`
Parameter for ComplexHeatmap::Heatmap.

`data` A data frame containing elements requested for annotation.

`left_bar` Named list of strings, where the strings are features in df that should be used for a barplot annotation on the left of the plot and the names are the names that will be used to caption the plots and their legends.

`right_bar` See left_bar.

`top_bar` See left_bar.

`bottom_bar` See left_bar.

`left_hm` Like left_bar, but with a heatmap annotation instead of a barplot annotation.

`right_hm` See left_hm.

`top_hm` See left_hm.

`bottom_hm` See left_hm.

`annotation_colours` Named list of heatmap annotations and their colours.

`min_colour` Colour used for the lowest value in the heatmap.

`max_colour` Colour used for the highest value in the heatmap.

`split_vector` A vector of partition indices.

`row_split` Standard parameter of `ComplexHeatmap::Heatmap`.

`column_split` Standard parameter of `ComplexHeatmap::Heatmap`.

`...` Additional parameters passed into ComplexHeatmap::Heatmap.

Value

Returns a heatmap (class "Heatmap" from package ComplexHeatmap) that displays the similarities between observations in the provided matrix.

Examples

```
#my_dl <- data_list(
#    list(
#        data = expression_df,
#        name = "expression_data",
#        domain = "gene_expression",
#        type = "continuous"
#    ),
#    list(
#        data = methylation_df,
#        name = "methylation_data",
#        domain = "gene_methylation",
#        type = "continuous"
#    ),
#    uid = "patient_id"
#)
#
#sc <- snf_config(my_dl, n_solutions = 10)
#
#sol_df <- batch_snf(my_dl, sc, return_sim_mats = TRUE)
#
#sim_mats <- sim_mats_list(sol_df)
#
#similarity_matrix_heatmap(
#    sim_mats[[1]],
#    cluster_solution = sol_df[1, ]
#)
```

sim_mats_list *Create or extract a sim_mats_list class object*

Description

Create or extract a `sim_mats_list` class object

Usage

```
sim_mats_list(x)
```

Arguments

`x`	The object to create or extract a `sim_mats_list` from.

Value

A `sim_mats_list` class object.

siw_euclidean_distance *Squared (including weights) Euclidean distance*

Description

Squared (including weights) Euclidean distance

Usage

```
siw_euclidean_distance(df, weights_row)
```

Arguments

`df`	data frame containing at least 1 data column.
`weights_row`	Single-row data frame where the column names contain the column names in df and the row contains the corresponding weights.

Value

distance_matrix A distance matrix.

snf_config *Define configuration for generating a set of SNF-based cluster solutions*

Description

`snf_config()` constructs an SNF config object which inherits from classes `snf_config` and `list`. This object is used to store all settings required to transform data stored in a `data_list` class object into a space of cluster solutions by SNF. The SNF config object contains the following components: 1. A settings data frame (inherits from `settings_df` and `data.frame`). Data frame that stores SNF-specific hyperparameters and information about feature selection and weighting, SNF schemes, clustering algorithms, and distance metrics. Each row of the settings data frame corresponds to a distinct cluster solution. 2. A clustering algorithms list (inherits from `clust_fns_list` and `list`), which stores all clustering algorithms that the settings data frame can point to. 3. A distance metrics list (inherits from `dist_metrics_list` and `list`), which stores all distance metrics that the settings data frame can point to. 4. A weights matrix (inherits from `weights_matrix`, `matrix`, and `array`'), which stores the feature weights to use prior to distance calculations. Each column of the weights matrix corresponds to a different feature in the data list and each row corresponds to a different row in the settings data frame.

Usage

```
snf_config(
  dl = NULL,
  sdf = NULL,
  dfl = NULL,
  cfl = NULL,
  wm = NULL,
  n_solutions = 0,
  min_removed_inputs = 0,
  max_removed_inputs = length(dl) - 1,
  dropout_dist = "exponential",
  min_alpha = NULL,
  max_alpha = NULL,
  min_k = NULL,
  max_k = NULL,
  min_t = NULL,
  max_t = NULL,
  alpha_values = NULL,
  k_values = NULL,
  t_values = NULL,
  possible_snf_schemes = c(1, 2, 3),
  clustering_algorithms = NULL,
  continuous_distances = NULL,
  discrete_distances = NULL,
  ordinal_distances = NULL,
  categorical_distances = NULL,
  mixed_distances = NULL,
  snf_input_weights = NULL,
  snf_domain_weights = NULL,
  retry_limit = 10,
  cnt_dist_fns = NULL,
  dsc_dist_fns = NULL,
  ord_dist_fns = NULL,
  cat_dist_fns = NULL,
  mix_dist_fns = NULL,
  automatic_standard_normalize = FALSE,
  use_default_dist_fns = FALSE,
  clust_fns = NULL,
  use_default_clust_fns = FALSE,
  weights_fill = "ones"
)
```

Arguments

`dl` A nested list of input data from `data_list()`.

`sdf` A `settings_df` class object. Overrides settings data frame related parameters.

`dfl` A `dist_fns_list` class object. Overrides distance functions list related parameters.

`cfl` A `clust_fns_list` class object. Overrides clustering functions list related parameters.

`wm` A `weights_matrix` class object. Overrides weights matrix related parameters.

`n_solutions` Number of rows to generate for the settings data frame.

`min_removed_inputs`
The smallest number of input data frames that may be randomly removed. By default, 0.

`max_removed_inputs`
The largest number of input data frames that may be randomly removed. By default, this is 1 less than all the provided input data frames in the data list.

`dropout_dist` Parameter controlling how the random removal of input data frames should occur. Can be "none" (no input data frames are randomly removed), "uniform" (uniformly sample between min_removed_inputs and max_removed_inputs to determine number of input data frames to remove), or "exponential" (pick number of input data frames to remove by sampling from min_removed_inputs to max_removed_inputs with an exponential distribution; the default).

`min_alpha` The minimum value that the alpha hyperparameter can have. Random assigned value of alpha for each row will be obtained by uniformly sampling numbers between `min_alpha` and `max_alpha` at intervals of 0.1. Cannot be used in conjunction with the `alpha_values` parameter.

`max_alpha` The maximum value that the alpha hyperparameter can have. See `min_alpha` parameter. Cannot be used in conjunction with the `alpha_values` parameter.

`min_k` The minimum value that the k hyperparameter can have. Random assigned value of k for each row will be obtained by uniformly sampling numbers between `min_k` and `max_k` at intervals of 1. Cannot be used in conjunction with the `k_values` parameter.

`max_k` The maximum value that the k hyperparameter can have. See `min_k` parameter. Cannot be used in conjunction with the `k_values` parameter.

`min_t` The minimum value that the t hyperparameter can have. Random assigned value of t for each row will be obtained by uniformly sampling numbers between `min_t` and `max_t` at intervals of 1. Cannot be used in conjunction with the `t_values` parameter.

`max_t` The maximum value that the t hyperparameter can have. See `min_t` parameter. Cannot be used in conjunction with the `t_values` parameter.

`alpha_values` A number or numeric vector of a set of possible values that alpha can take on. Value will be obtained by uniformly sampling the vector. Cannot be used in conjunction with the `min_alpha` or `max_alpha` parameters.

`k_values` A number or numeric vector of a set of possible values that k can take on. Value will be obtained by uniformly sampling the vector. Cannot be used in conjunction with the `min_k` or `max_k` parameters.

`t_values` A number or numeric vector of a set of possible values that t can take on. Value will be obtained by uniformly sampling the vector. Cannot be used in conjunction with the `min_t` or `max_t` parameters.

`possible_snf_schemes`

A vector containing the possible snf_schemes to uniformly randomly select from. By default, the vector contains all 3 possible schemes: c(1, 2, 3). 1 corresponds to the "individual" scheme, 2 corresponds to the "domain" scheme, and 3 corresponds to the "twostep" scheme.

`clustering_algorithms`

A list of clustering algorithms to uniformly randomly pick from when clustering. When not specified, randomly select between spectral clustering using the eigen-gap heuristic and spectral clustering using the rotation cost heuristic. See ?clust_fns_list for more details on running custom clustering algorithms.

`continuous_distances`

A vector of continuous distance metrics to use when a custom dist_fns_list is provided.

`discrete_distances`

A vector of categorical distance metrics to use when a custom dist_fns_list is provided.

`ordinal_distances`

A vector of categorical distance metrics to use when a custom dist_fns_list is provided.

`categorical_distances`

A vector of categorical distance metrics to use when a custom dist_fns_list is provided.

`mixed_distances`

A vector of mixed distance metrics to use when a custom dist_fns_list is provided.

`snf_input_weights`
Nested list containing weights for when SNF is used to merge individual input measures (see ?generate_snf_weights)

`snf_domain_weights`
Nested list containing weights for when SNF is used to merge domains (see ?generate_snf_weights)

`retry_limit` The maximum number of attempts to generate a novel row. This function does not return matrices with identical rows. As the range of requested possible settings tightens and the number of requested rows increases, the risk of randomly generating a row that already exists increases. If a new random row has matched an existing row `retry_limit` number of times, the function will terminate.

`cnt_dist_fns` A named list of continuous distance metric functions.

`dsc_dist_fns` A named list of discrete distance metric functions.

`ord_dist_fns` A named list of ordinal distance metric functions.

`cat_dist_fns` A named list of categorical distance metric functions.

`mix_dist_fns` A named list of mixed distance metric functions.

`automatic_standard_normalize`
If TRUE, will automatically use standard normalization prior to calculation of any numeric distances. This parameter overrides all other distance functions list-related parameters.

`use_default_dist_fns`
If TRUE, prepend the base distance metrics (euclidean distance for continuous, discrete, and ordinal data and gower distance for categorical and mixed data) to the resulting distance metrics list.

`clust_fns` A list of named clustering functions

`use_default_clust_fns`
If TRUE, prepend the base clustering algorithms (spectral_eigen and spectral_rot, which apply spectral clustering and use the eigen-gap and rotation cost heuristics respectively for determining the number of clusters in the graph) to clust_fns.

`weights_fill` String indicating what to populate generate rows with. Can be "ones" (default; fill matrix with 1), "uniform" (fill matrix with uniformly distributed random values), or "exponential" (fill matrix with exponentially distributed random values).

Value

An `snf_config` class object.

Examples

```
# Simple random config for 5 cluster solutions
input_dl <- data_list(
    list(anxiety, "anxiety", "behaviour", "ordinal"),
    list(depress, "depressed", "behaviour", "ordinal"),
    uid = "unique_id"
)
my_sc <- snf_config(
    dl = input_dl,
    n_solutions = 5
)

# specifying possible K range
my_sc <- snf_config(
    dl = input_dl,
    n_solutions = 5,
    min_k = 20,
    max_k = 40
)

# Random feature weights across from uniform distribution
my_sc <- snf_config(
    dl = input_dl,
    n_solutions = 5,
    min_k = 20,
    max_k = 40,
    weights_fill = "uniform"
)

# Specifying custom pre-built clustering and distance functions
# - Random alternation between 2-cluster and 5-cluster solutions
# - When continuous or discrete data frames are being processed,
#   randomly alternate between standardized/normalized Euclidean
#   distance vs. regular Euclidean distance
my_sc <- snf_config(
    dl = input_dl,
    n_solutions = 5,
    min_k = 20,
    max_k = 40,
    weights_fill = "uniform",
    clust_fns = list(
        "two_cluster_spectral" = spectral_two,
        "five_cluster_spectral" = spectral_five
    ),
    cnt_dist_fns = list(
         "euclidean" = euclidean_distance,
         "std_nrm_euc" = sn_euclidean_distance
    ),
    dsc_dist_fns = list(
         "euclidean" = euclidean_distance,
         "std_nrm_euc" = sn_euclidean_distance
    )
)
```

split_parser *Helper function to determine which row and columns to split on*

Description

Helper function to determine which row and columns to split on

Usage

```
split_parser(
  row_split_vector = NULL,
  column_split_vector = NULL,
  row_split = NULL,
  column_split = NULL,
  n_rows,
  n_columns
)
```

Arguments

`row_split_vector`
A vector of row indices to split on.

`column_split_vector`
A vector of column indices to split on.

`row_split` Standard parameter of `ComplexHeatmap::Heatmap`.

`column_split` Standard parameter of `ComplexHeatmap::Heatmap`.

`n_rows` The number of rows in the data.

`n_columns` The number of columns in the data.

Value

"list"-class object containing row_split and column_split character vectors to pass into ComplexHeatmap::Heatmap.

`subc_v` *Mock ABCD subcortical volumes data*

Description

Like the mock data frame "abcd_subc_v", but with "unique_id" as the "uid".

Usage

```
subc_v
```

Format

`subc_v`**:**

A data frame with 174 rows and 31 columns:

unique_id The unique identifier of the ABCD dataset

... Subcortical volumes of various ROIs (mm^3, I think)

Source

Though this data is no longer "real" ABCD data, the reference for using ABCD as a data source is below:

Data used in the preparation of this article were obtained from the Adolescent Brain Cognitive DevelopmentSM (ABCD) Study (https://abcdstudy.org), held in the NIMH Data Archive (NDA). This is a multisite, longitudinal study designed to recruit more than 10,000 children age 9-10 and follow them over 10 years into early adulthood. The ABCD Study® is supported by the National Institutes of Health and additional federal partners under award numbers U01DA041048, U01DA050989, U01DA051016, U01DA041022, U01DA051018, U01DA051037, U01DA050987, U01DA041174, U01DA041106, U01DA041117, U01DA041028, U01DA041134, U01DA050988, U01DA051039, U01DA041156, U01DA041025, U01DA041120, U01DA051038, U01DA041148, U01DA041093, U01DA041089, U24DA041123, U24DA041147. A full list of supporters is available at https://abcdstudy.org/federal-partners.html. A listing of participating sites and a complete listing of the study investigators can be found at https://abcdstudy.org/consortium_members/. ABCD consortium investigators designed and implemented the study and/or provided data but did not necessarily participate in the analysis or writing of this report. This manuscript reflects the views of the authors and may not reflect the opinions or views of the NIH or ABCD consortium investigators.

subsample_dl *Create subsamples of a data list*

Description

Given a data list, return a list of smaller data lists that are generated through random sampling (without replacement). The results of this function can be passed into `batch_snf_subsamples()` to obtain a list of resampled solutions data frames.

Usage

```
subsample_dl(
  dl,
  n_subsamples,
  subsample_fraction = NULL,
  n_observations = NULL
)
```

Arguments

`dl`	A nested list of input data from `data_list()`.
`n_subsamples`	Number of subsamples to create.
`subsample_fraction`	Percentage of patients to include per subsample.
`n_observations`	Number of patients to include per subsample.

Value

A "list" class object containing n_subsamples number of data lists. Each of those data lists contains a random subsample_fraction fraction of the observations of the provided data list.

Examples

```
# my_dl <- data_list(
#     list(subc_v, "subcortical_volume", "neuroimaging", "continuous"),
#     list(income, "household_income", "demographics", "continuous"),
#     list(pubertal, "pubertal_status", "demographics", "continuous"),
#     uid = "unique_id"
# )
#
# my_dl_subsamples <- subsample_dl(
#     my_dl,
#     n_subsamples = 20,
#     subsample_fraction = 0.85
# )
```

subsample_pairwise_aris

Calculate pairwise adjusted Rand indices across subsamples of data

Description

Given a list of subsampled solutions data frames from 'batch_snf_subsamples(), this function calculates the adjusted Rand indices across all the subsamples of each solution. ARI calculation between two subsamples only factors in observations that were present in both subsamples.

Usage

```
subsample_pairwise_aris(subsample_solutions, verbose = FALSE)
```

Arguments

subsample_solutions
A list of solutions data frames from subsamples of the data. This object is generated by the function batch_snf_subsamples().

verbose
If TRUE, output progress to console.

Value

A two-item list: "raw_aris", a list of inter-subsample pairwise ARI matrices (one for each full cluster solution) and "ari_summary", a data frame containing the mean and SD of the inter-subsample ARIs for each original cluster solution.

Examples

```
# my_dl <- data_list(
#     list(subc_v, "subcortical_volume", "neuroimaging", "continuous"),
#     list(income, "household_income", "demographics", "continuous"),
#     list(pubertal, "pubertal_status", "demographics", "continuous"),
#     uid = "unique_id"
# )
#
# sc <- snf_config(my_dl, n_solutions = 5, max_k = 40)
#
# my_dl_subsamples <- subsample_dl(
#     my_dl,
#     n_subsamples = 20,
#     subsample_fraction = 0.85
# )
#
# batch_subsample_results <- batch_snf_subsamples(
#     my_dl_subsamples,
#     sc,
#     verbose = TRUE
# )
#
# pairwise_aris <- subsample_pairwise_aris(
#     batch_subsample_results
#     verbose = TRUE
# )
#
# # Visualize ARIs
# ComplexHeatmap::Heatmap(
#     pairwise_aris[[1]],
#     heatmap_legend_param = list(
#         color_bar = "continuous",
#         title = "Inter-Subsample\nARI",
#         at = c(0, 0.5, 1)
#     ),
#     show_column_names = FALSE,
#     show_row_names = FALSE
# )
```

summarize_clust_fns_list

Summarize a clust_fns_list object

Description

Summarize a clust_fns_list object

Usage

```
summarize_clust_fns_list(cfl)
```

Arguments

`cfl` A `clust_fns_list` class object.

Value

summary_df "data.frame" class object containing the name and index of each clustering algorithm in te provided `clust_fns_list`.

`summarize_dfl` *Summarize metrics contained in a dist_fns_list*

Description

Summarize metrics contained in a dist_fns_list

Usage

```
summarize_dfl(dist_fns_list)
```

Arguments

`dist_fns_list` A dist_fns_list.

Value

"data.frame"-class object summarizing items in a distance metrics list.

`summarize_dl` *Summarize a data list*

Description

[Deprecated] Defunct function for summarizing a data list. Please use `summary()` instead.

Usage

```
summarize_dl(data_list, scope = "component")
```

Arguments

`data_list` A nested list of input data from `data_list()`.

`scope` The level of detail for the summary. Options are:

- "component" (default): One row per component (data frame) in the data list.
- "feature": One row for each feature in the data list.

Value

data.frame class object summarizing all components (or features if scope == "component").

summary.data_list *Summary method for class* data_list

Description

Returns a data list summary (data.frame class object) containing information on components, features, variable types, domains, and component dimensions.

Usage

```
## S3 method for class 'data_list'
summary(object, scope = "component", ...)
```

Arguments

object	A data_list class object.
scope	The level of detail for the summary. By default, this is set to "component", which returns a summary of the data list at the component level. Can also be set to "feature", resulting in a summary at the feature level.
...	Other arguments passed to summary (not used in this function)

Value

A data.frame class object. If scope is "component", each row shows the name, variable type, domain, and dimensions of each component. If scope is "feature", each row shows the name, variable type, and domain of each feature.

summary_features *Pull features used to calculate summary p-values from an object*

Description

Pull features used to calculate summary p-values from an object

Usage

```
summary_features(x)
```

Arguments

x	The object to extract summary features from.

Value

A character vector of summary features.

`train_test_assign` *Training and testing split*

Description

Given a vector of uid_id and a threshold, returns a list of which members should be in the training set and which should be in the testing set. The function relies on whether or not the absolute value of the Jenkins's one_at_a_time hash function exceeds the maximum possible value (2147483647) multiplied by the threshold.

Usage

```
train_test_assign(train_frac, uids, seed = 42)
```

Arguments

`train_frac`	The fraction (0 to 1) of observations for training
`uids`	A character vector of UIDs to be distributed into training and test sets.
`seed`	Seed used for Jenkins's one_at_a_time hash function.

Value

A named list containing the training and testing uid_ids.

`uids` *Pull UIDs from an object*

Description

Pull UIDs from an object

Usage

```
uids(x)
```

Arguments

`x`	The object to extract UIDs from.

Value

A character vector of UIDs.

validate_solutions_df *Validator for* solutions_df *class object*

Description

Validator for solutions_df class object

Usage

```
validate_solutions_df(sol_dfl)
```

Arguments

sol_dfl	A solutions data frame-like object to be validated and converted into a solutions data frame.

Value

If sol_dfl has a valid structure for a solutions_df class object, returns the input unchanged. Otherwise, raises an error.

var_manhattan_plot *Manhattan plot of feature-feature association p-values*

Description

Manhattan plot of feature-feature association p-values

Usage

```
var_manhattan_plot(
  dl,
  key_var,
  neg_log_pval_thresh = 5,
  threshold = NULL,
  point_size = 5,
  text_size = 20,
  plot_title = NULL,
  hide_x_labels = FALSE,
  bonferroni_line = FALSE
)
```

Arguments

`dl` List of data frames containing data information.

`key_var` Feature for which the association p-values of all other features are plotted.

`neg_log_pval_thresh` Threshold for negative log p-values.

`threshold` p-value threshold to plot dashed line at.

`point_size` Size of points in the plot.

`text_size` Size of text in the plot.

`plot_title` Title of the plot.

`hide_x_labels` If TRUE, hides x-axis labels.

`bonferroni_line` If TRUE, plots a dashed black line at the Bonferroni-corrected equivalent of the p-value threshold.

Value

A Manhattan plot (class "gg", "ggplot") showing the association p-values of features against one key feature in a data list.

Examples

```
dl <- data_list(
    list(subc_v, "subcortical_volume", "neuroimaging", "continuous"),
    list(income, "household_income", "demographics", "continuous"),
    list(pubertal, "pubertal_status", "demographics", "continuous"),
    list(anxiety, "anxiety", "behaviour", "ordinal"),
    list(depress, "depressed", "behaviour", "ordinal"),
    uid = "unique_id"
)

var_manhattan <- var_manhattan_plot(
    dl,
    key_var = "household_income",
    plot_title = "Correlation of Features with Household Income",
    text_size = 16,
    neg_log_pval_thresh = 3,
    threshold = 0.05
)
```

weights_matrix *Generate a matrix to store feature weights*

Description

Function for building a weights matrix independently of an SNF config. The weights matrix contains one row corresponding to each row of the settings data frame in an SNF config (one row for each resulting cluster solution) and one column for each feature in the data list used for clustering. Values of the weights matrix are passed to distance metrics functions during the conversion of input data frames to distance matrices. Typically, there is no need to use this function directly. Instead, users should provide weights matrix-building parameters to the `snf_config()` function.

Usage

```
weights_matrix(dl = NULL, n_solutions = 1, weights_fill = "ones")
```

Arguments

`dl`	A nested list of input data from `data_list()`.
`n_solutions`	Number of rows to generate the template weights matrix for.
`weights_fill`	String indicating what to populate generate rows with. Can be "ones" (default; fill matrix with 1), "uniform" (fill matrix with uniformly distributed random values), or "exponential" (fill matrix with exponentially distributed random values).

Value

wm A properly formatted matrix containing columns for all the features that require weights and rows.

Examples

```
input_dl <- data_list(
    list(subc_v, "subcortical_volume", "neuroimaging", "continuous"),
    list(income, "income", "demographics", "continuous"),
    list(pubertal, "pubertal_status", "demographics", "continuous"),
    uid = "unique_id"
)

sc <- snf_config(input_dl, n_solutions = 5)

wm <- weights_matrix(input_dl, n_solutions = 5, weights_fill = "uniform")

# updating an SNF config in parts
sc$"weights_matrix" <- wm
```

Index