Package 'PersonalityTypes'

July 2, 2025

Type Package

Title A Gaussian Mixture Model Approach to Analyzing Personality Types in Five-Factor Model Data

Version 0.1

Description This pipeline begins with preprocessing raw survey data and performing Exploratory Factor Analysis (EFA) to derive latent factor scores. It then applies Gaussian Mixture Models (GMM) to these scores, using the Normalized Entropy Criterion (NEC) and Normalized Variation of Information (NVI) to identify the optimal and most stable cluster solution, representing distinct personality types. The package includes methods for validating cluster robustness via permutation-based enrichment testing. Finally, it facilitates the interpretation of these types by linking them to external outcome variables through Compositional Regression, which correctly handles cluster membership probabilities as compositional predictors. A suite of specialized visualization functions is provided for each stage of the analysis.

Imports dplyr, tidyr, ggplot2, gtools, ggpubr, mclust

Depends R (>= 4.0)

NeedsCompilation yes

LazyData true

License GPL (>= 3)

BugReports https://github.com/statpng/PersonalityTypes/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Maintainer Kipoong Kim <kkp7700@gmail.com>

Contents

`analysis.DataPreprocessing`

Preprocess Personality Survey Data

Description

This function preprocesses a data frame containing personality survey data, specifically tailored for a structure where items from different factors (e.g., Big Five "OCEAN" traits) are in separate blocks.

Usage

```
analysis.DataPreprocessing(df, FactorNames = strsplit("OCEAN", "")[[1]])
```

Arguments

`df`	A data frame containing the raw survey data. It is expected to have columns for personality items (e.g., `A1`, `C1`...) and corresponding demographic blocks (e.g., `age_A`, `gender_A`). See the function's code comments for the expected structure.
`FactorNames`	A character vector of single-letter prefixes used to identify the personality factor items. Defaults to `c("O", "C", "E", "A", "N")` for the Big Five model.

Details

This function is designed to clean and restructure a specific type of psychological survey data before statistical modeling. The primary goal is to extract a clean numeric matrix of personality items (`X`) and a corresponding processed data frame (`df`).

The key preprocessing steps are:

1. **Filtering for Complete Responses**: The function first identifies all columns corresponding to the personality items (e.g., `O1`, `C1`, `E1`, etc.) based on the `FactorNames` argument. It then filters the data frame to retain only the rows (participants) that have no missing values (`NA`) across *all* of these item columns. This ensures that the subsequent analyses are performed on a complete dataset. This step is statistically important as many multivariate models (like Factor Analysis or GMM) require complete data.
2. **Harmonizing Demographic Data**:
   - `age_min`: It calculates a single age value for each participant by taking the minimum of all available age columns (e.g., `age_O`, `age_C`, etc.). This resolves potential discrepancies.

- gender_vote: It creates a single, consistent gender variable. For each participant, it inspects all gender columns (e.g., gender_O, gender_C) and assigns the most frequently occurring value. This "voting" mechanism is a robust method for data harmonization. The resulting variable is then converted to an English factor (male, female).

3. **Extracting the Analysis Matrix (X)**: A final matrix X is created containing only the numeric responses to the personality items. This matrix is the primary input for subsequent multivariate statistical analyses like clustering or factor analysis.

The function also prints a summary of the resulting matrix X to the console, including its dimensions and the count of items per factor, which serves as a quick diagnostic check.

Value

A list containing three elements:

ID	A vector of participant IDs corresponding to the cleaned data.
X	A numeric data frame or matrix containing only the cleaned personality item data, suitable for modeling.
df	The preprocessed version of the input data frame, including the harmonized age_min and gender_vote columns and filtered to match the rows in X.

---

analysis.Enrichment *Perform and Visualize Cluster Enrichment Analysis*

Description

A wrapper function that runs the enrichment analysis, identifies significant clusters, and creates a diagnostic plot.

Usage

```
analysis.Enrichment(
  fit.GMM,
  Xclust,
  Xnew,
  seed,
  n.clust = NULL,
  nrep = 100,
  filename = NULL
)
```

Arguments

fit.GMM	The fitted GMM object from mclust::Mclust.
Xclust	A data frame containing the data with a class column for cluster assignments.
Xnew	The raw numeric data matrix used for the GMM.
seed	The random seed used in the original GMM fit, passed to fit.enrichment for reproducibility.
n.clust	The number of clusters in the model.
nrep	The number of permutations for the enrichment test.
filename	A base string for the output PDF plot file.

Details

This function orchestrates the cluster validation workflow. It takes a fitted GMM result and performs the following steps:

1. Calls `fit.enrichment` to conduct the permutation-based statistical test for cluster robustness.
2. Generates a diagnostic "volcano-like" plot, plotting the log-10 p-value against the relative enrichment score (ρ_{rel}). This plot visually separates significant clusters (typically in the upper right, with high enrichment and low p-value) from non-significant ones.
3. Identifies significant clusters (`SigCluster`) based on a significance threshold (e.g., p-value < 0.05).
4. Returns a list containing the enrichment results and other relevant model objects for downstream use.

The function also contains commented-out code to demonstrate how one might use `reassign.enrichment` to create a new, consolidated set of cluster assignments.

Value

A list containing:

`Xnew`	The original data matrix.
`Xclust2`	The original `Xclust` data frame (note: reassignment is commented out in the provided code).
`fit.GMM`	The original fitted GMM object.
`res.enrichment`	The list object returned by `fit.enrichment`.

See Also

`fit.enrichment`, `reassign.enrichment`

`analysis.Enrichment.range`

Perform Enrichment Analysis Across a Range of Cluster Numbers

---

Description

Systematically runs the permutation-based enrichment analysis for a series of Gaussian Mixture Models with different numbers of clusters.

Usage

```
analysis.Enrichment.range(Xnew, n.clust = 10, nrep = 100, seed)
```

Arguments

`Xnew`	A numeric matrix or data frame of observations.
`n.clust`	The maximum number of clusters to test. The analysis will be performed for `k = 2, 3, ..., n.clust`.
`nrep`	The number of permutations to perform in each call to `fit.enrichment`.
`seed`	A random seed to ensure that the initial GMM fit is reproducible for each number of clusters.

Details

This function serves as a sensitivity analysis tool for cluster robustness. Instead of evaluating a single clustering solution, it iterates through a range of cluster numbers (from 2 to `n.clust`).

For each number of clusters `k` in the range, the function performs the following two steps:

1. Fits a Gaussian Mixture Model with `k` components to the data (`mclust::Mclust`).
2. Immediately runs the statistical validation procedure (`fit.enrichment`) on that `k`-cluster model.

The results of each validation are stored in a list. This allows a user to compare not only the number of significant clusters but also their enrichment scores and p-values across different model complexities (i.e., different `k`). This approach helps in making a more informed decision on the final number of clusters by assessing how the statistical significance of cluster profiles changes as `k` is varied.

Value

A list of lists. The k-th element of the returned list (`result[[k]]`) contains the full output from `fit.enrichment` for the k-cluster model. Elements for $k < 2$ will be `NULL`.

See Also

`fit.enrichment`, `analysis.Enrichment`

`analysis.FA` *Perform First and Second-Order Factor Analysis*

Description

Conducts a comprehensive factor analysis workflow, including data suitability tests, determination of the number of factors, fitting a first-order model, and optionally fitting a second-order (hierarchical) model.

Usage

```
analysis.FA(X, ID, nfactors = NULL, nfactors2 = NULL, filename = NULL)
```

Arguments

`X`	A numeric matrix or data frame of the observed variables (e.g., item responses).
`ID`	A vector of identifiers for each row in `X`.
`nfactors`	An optional integer specifying the number of first-order factors. If `NULL` (the default), it is determined automatically via parallel analysis.
`nfactors2`	An optional integer specifying the number of second-order factors. If `NULL` (the default), the second-order analysis is skipped.
`filename`	A base string for the output PDF file names (e.g., scree plots, diagrams).

Details

This function automates the key steps of Exploratory Factor Analysis (EFA), a statistical method used to uncover the underlying latent structure (factors) from a set of observed variables.

The workflow consists of three main parts:

1. **Assessing Factorability**: Before fitting a model, the function assesses whether the data is suitable for factor analysis using two standard statistical tests:
   - **Kaiser-Meyer-Olkin (KMO) Test**: Calculates the Measure of Sampling Adequacy (MSA). The KMO index represents the ratio of the squared correlation between variables to the squared partial correlation between variables. A value of 0.6 or higher is generally considered acceptable for factor analysis.
   - **Bartlett's Test of Sphericity**: This tests the null hypothesis that the correlation matrix is an identity matrix (i.e., variables are unrelated). A significant p-value (e.g., < 0.05) is required, indicating that the variables are sufficiently intercorrelated to be factored.
2. **Determining the Number of Factors**: The function uses Parallel Analysis (`psych::fa.parallel`), a robust method for determining the number of factors to retain. This method compares the eigenvalues of the actual data to the eigenvalues from simulated random data. Factors are retained if their eigenvalues are greater than those from the corresponding random data. A scree plot is generated to visualize this. If `nfactors` is not specified, the function defaults to using the Kaiser criterion (retaining factors with eigenvalues > 1).
3. **Model Fitting and Score Extraction**:
   - **First-Order Model**: A primary factor model is fitted using Principal Axis Factoring (`fm="pa"`), which is a common EFA extraction method. An oblique rotation (`rotate="oblimin"`) is used, which allows the extracted factors to be correlated. This is often theoretically appropriate in social sciences. The function then calculates factor scores, which are estimates of each individual's standing on the latent factors.
   - **Second-Order Model (Optional)**: If `nfactors2` is specified, a hierarchical factor analysis is performed using `psych::fa.multi`. This technique treats the first-order factors as indicators to uncover a higher-level, more general latent structure. The final `Xnew` object will then contain the scores for these second-order factors.

The function saves diagnostic plots (scree plot, factor diagrams) as PDF files and returns the final factor scores, which can be used as input for subsequent analyses like clustering or regression.

Value

A list containing:

`ID`	The input identifier vector.
`X`	The original input data matrix.
`Xnew`	A matrix of the final factor scores. This will be the first-order factor scores if `nfactors2` is NULL, or the second-order factor scores otherwise.
`console`	A placeholder for console output (currently not used).

See Also

`fa`, `fa.parallel`, `fa.multi`, `KMO`

`analysis.GMM.best` *Find the Best Gaussian Mixture Model using NEC*

Description

Performs GMM clustering for a range of cluster numbers and random seeds, selecting the best model based on the Normalized Entropy Criterion (NEC).

Usage

```
analysis.GMM.best(
  Xnew = NULL,
  ID = NULL,
  max.clust = 10,
  filename = NULL,
  subsample.size = 100,
  print.pdf = TRUE,
  q = 1,
  type = "NEC"
)
```

Arguments

`Xnew`	A numeric matrix or data frame of observations.
`ID`	An optional identifier for the analysis run.
`max.clust`	The maximum number of clusters to test. Default is 10.
`filename`	A base string for output PDF file names. If `NULL`, plots are not saved.
`subsample.size`	The number of random seeds to try. Default is 100.
`type`	The criterion for model selection (currently hardcoded to "NEC").

Details

This function systematically explores the solution space of Gaussian Mixture Models to find the optimal number of clusters. It iterates through a specified number of clusters (`1:max.clust`) and a set of random initializations (`seed.set`).

For each combination of cluster number and seed, it fits a GMM (`mclust::Mclust` with `modelNames="VVV"`) and calculates the NEC using `calculate_nec`. The `q` value for Tsallis entropy within NEC is hard-coded to 1.5 in this implementation.

The function identifies the combination of cluster number and seed that yields the minimum NEC value globally. It generates several diagnostic plots:

1. A plot of NEC values across all tested seeds, highlighting the best model.
2. A plot of sorted NEC values for the optimal number of clusters.
3. Boxplots of the data distributions for the top 3 best clustering results.

The function returns a comprehensive list of results, including the matrix of all NEC values and the best model found.

Value

A list containing:

`ID`	The analysis identifier.
`Xnew`	The input data.
`NEC.mat`	A matrix of NEC values (rows are seeds, columns are cluster numbers).
`fit.best`	The `Mclust` object for the best model found (one of the top 3).
`wh.best`	A vector with the optimal number of clusters and the corresponding seed.
`Xclust`	The input data with an appended `class` column from the best model.

`analysis.NVI` *Analyze Clustering Stability using Normalized Variation of Information (NVI)*

Description

For a fixed number of clusters, this function assesses the stability of clustering results from different random initializations using NVI.

Usage

```
analysis.NVI(Xnew, nclust, NEC.mat, top = 20, filename = NULL)
```

Arguments

`Xnew`	The numeric matrix or data frame of observations.
`nclust`	The specific number of clusters to analyze for stability.
`NEC.mat`	The matrix of NEC values produced by `analysis.GMM.best`.
`top`	The number of top-performing runs (by NEC) to compare. Default is 20.
`filename`	A base string for output PDF file names.

Details

After determining a candidate for the optimal number of clusters (`nclust`), this function evaluates the stability of the clustering solution. It selects the top `top` runs (i.e., seeds) for that `nclust` based on the lowest NEC values from `NEC.mat`.

It then computes the pairwise Normalized Variation of Information (NVI) between the partitions from these top runs. NVI is a measure of distance between two clusterings, with lower values indicating higher similarity.

The function calculates the average NVI for each run against all other top runs. The run with the minimum average NVI is considered the most stable and representative clustering solution.

It generates plots to visualize the stability analysis:

1. A line plot of the averaged NVI values for each tested seed.
2. Boxplots for the clusterings, ordered by their stability (from most to least stable).

Value

A list containing:

VI_matrix	The pairwise NVI matrix between the top runs.
Xclust.NVI	The input data with an appended class column from the most stable model.
fit.best.NVI	The Mclust object for the most stable model.
wh.best.NVI	A vector with the number of clusters and the seed of the most stable model.

See Also

analysis.GMM.best

---

bootstrap_test *Bootstrap Confidence Intervals for Compositional Regression Coefficients*

Description

Constructs non-parametric confidence intervals for CompReg coefficients using bootstrapping.

Usage

```
bootstrap_test(X_screen, y, rep)
```

Arguments

X_screen	A numeric matrix of log-transformed compositional predictors.
y	A numeric vector for the response variable.
rep	The number of bootstrap replicates to perform.

Details

This function provides a robust method for estimating the uncertainty of the regression coefficients. It implements the standard case resampling bootstrap:

1. A bootstrap sample is created by drawing n observations with replacement from the original data (X, y).
2. A CompReg model is fitted to this bootstrap sample.
3. This process is repeated rep times to generate a distribution of the estimated coefficients and t-statistics.

The function calculates two types of 95% confidence intervals:

- **Percentile Interval**: The 2.5th and 97.5th percentiles of the bootstrapped coefficient distribution.
- **Studentized (Pivotal) Interval**: A more accurate interval that uses the distribution of the bootstrapped t-statistic. It is generally preferred for its better statistical properties.

Value

A list containing the original estimates (beta_hat, t_ref) and data frames for the percentile (per_int) and studentized (stud_int) confidence intervals.

`calculate_nec` *Calculate Normalized Entropy Criterion (NEC)*

Description

Calculates the Normalized Entropy Criterion for a given clustering model.

Usage

```
calculate_nec(model, base_loglik, q = 1)
```

Arguments

`model`	A model object, typically from `mclust::Mclust`, which must contain posterior probabilities (`model$z`) and log-likelihood (`model$loglik`).
`base_loglik`	The log-likelihood of the baseline model (e.g., G=1).
`q`	The entropic-index for the Tsallis entropy calculation. Defaults to 1 (Shannon Entropy).

Details

The NEC is used to select the optimal number of clusters. It balances the improvement in model fit (log-likelihood) against the uncertainty in classification (entropy). The NEC is defined as:

$$NEC(K) = \frac{E(K)}{\log L(K) - \log L(1)}$$

where K is the number of clusters.

- $E(K)$ is the total classification entropy, calculated by summing the entropy of the posterior probability distribution for each observation. This function uses Tsallis entropy for this calculation: $E(K) = \sum_{i=1}^{N} S_q(z_i)$, where z_i is the vector of posterior probabilities for observation i.
- $\log L(K)$ is the log-likelihood of the model with K clusters.
- $\log L(1)$ is the log-likelihood of the baseline model with one cluster. The optimal number of clusters is the one that minimizes the NEC.

Value

A single numeric value representing the NEC.

CompReg *Fit a Compositional Regression Model*

Description

Fits a linear regression model with compositional predictors using a constrained least-squares approach.

Usage

```
CompReg(X, y)
```

Arguments

X	A numeric matrix of log-transformed compositional predictors.
y	A numeric vector for the response variable.

Details

This function implements the compositional regression model as described in SMMR (Statistical Methods in Medical Research). Compositional predictors (X) are variables that represent proportions of a whole, and thus have a sum-to-k constraint (e.g., sum to 1). Standard OLS is not appropriate for such data.

This model uses log-transformed predictors ($X_{log} = \log(X)$) and estimates coefficients (β) under the constraint that their sum is zero: $\sum_{j=1}^{p} \beta_j = 0$. This constraint ensures model identifiability and allows for proper interpretation of the coefficients. The estimated model is:

$$y = \mu + X_{log}\beta + \epsilon \quad \text{subject to} \quad \sum \beta_j = 0$$

The coefficient β_j is interpreted as the effect of increasing the proportion of component j relative to the geometric mean of all components. The function uses a projection matrix approach to solve this constrained optimization problem and provides estimates for the intercept (μ), coefficients (β), standard errors (se), and t-statistics (t).

Value

A list containing:

mu	The estimated intercept.
beta	A named vector of the estimated regression coefficients.
t	A named vector of the t-statistics for each coefficient.
se	A named vector of the standard errors for each coefficient.
M, inv, Q, A	Matrices used in the constrained estimation process.

CompReg.forest *Create a Forest Plot for Compositional Regression Results*

Description

Visualizes the results of a compositional regression analysis, showing the estimated coefficients and their confidence intervals.

Usage

```
CompReg.forest(result, DV.name, class = NULL, width = 30)
```

Arguments

result	A single result object for one dependent variable, taken from the list returned by png.CompReg.
DV.name	A string with the name of the dependent variable for the plot title.
class	An optional vector of labels for the predictors (clusters).
width	An integer to adjust the spacing for the plot layout.

Details

This function takes the result from png.CompReg and creates a forest plot using the forestploter package. A forest plot is an effective way to display regression results, as it provides a clear visual summary of the magnitude, direction, and uncertainty (confidence interval) of the effect for each predictor. Each row in the plot corresponds to a compositional predictor, with a point estimate and a horizontal line representing the confidence interval. A vertical reference line at zero helps in quickly identifying statistically significant predictors (whose CIs do not cross zero).

Value

A forestploter plot object that can be printed to the graphics device.

fit.enrichment *Assess Cluster Enrichment using a Permutation Test*

Description

Performs a permutation-based statistical test to evaluate the enrichment (i.e., local density) of each cluster centroid from a GMM.

Usage

```
fit.enrichment(mod, BIC = NULL, n.clust = NULL, nrep = 100, seed)
```

Arguments

mod	The fitted model object from mclust::Mclust.
BIC	Optional. An mclustBIC object. If provided, Mclust is run on permuted data using this to select the number of clusters.
n.clust	The number of clusters to fit on the permuted data. Used if BIC is NULL.
nrep	The number of permutations to perform to build the null distribution.
seed	A random seed for reproducibility of the GMM fitting on permuted data.

Details

This function provides a statistical assessment of how "real" each cluster is. A robust cluster is expected to form in a region of high data density. This function tests whether the density at each observed cluster centroid is significantly higher than what would be expected by chance.

The statistical procedure is as follows:

1. **Observed Density**: For the original fitted model (mod), it calculates the probability density (ρ_{obs}) at the location of each cluster's centroid using the estimated GMM parameters.
2. **Permutation (Null Distribution)**: It generates a null distribution for the density. This is done by: a. Creating nrep permuted datasets. In each, the values within the data matrix are randomly shuffled, breaking any real structure. b. For each permuted dataset, a new GMM is fitted with the same specifications (number of clusters, model name). c. The density (ρ_{perm}) is calculated at the locations of the *original* centroids using the parameters from the model fitted to the *permuted* data.
3. **Enrichment and p-value**:
   - The relative enrichment (ρ_{rel}) for each cluster is the ratio of the observed density to the mean of the densities from the null distribution: $\rho_{rel} = \rho_{obs}/\text{mean}(\rho_{perm})$. A value > 1 suggests the cluster is in a denser-than-random region.
   - An empirical p-value is calculated for each cluster as the proportion of permutations where the permuted density was greater than or equal to the observed density. This represents the probability of observing such a high density by chance alone. A low p-value suggests the cluster is statistically significant.

Value

A list containing:

rho_obs	A vector of the observed densities at each original cluster centroid.
rho_perm	A matrix of densities from the permutation test (rows are permutations, columns are clusters).
rho_rel	A vector of the relative enrichment scores for each cluster.
pvalues	A vector of empirical p-values for each cluster.

NVI.BoxPlot *Create Boxplot for the Best NVI Result*

Description

A wrapper function that generates a boxplot for the most stable clustering solution identified by the `analysis.NVI` function.

Usage

```
NVI.BoxPlot(res.NVI, filename = NULL)
```

Arguments

`res.NVI`	A list object returned by the `analysis.NVI` function.
`filename`	A base string for the output PDF file name.

Details

This function simplifies the process of visualizing the most stable clustering profile. It extracts the necessary components (`wh.best.NVI`, `Xclust.NVI`) from the result object of `analysis.NVI` and passes them to the internal plotting function `Xclust.boxplot.filename_new` to generate a labeled boxplot.

Value

This function does not return a value. It saves a plot to a PDF file.

See Also

`analysis.NVI`, `Xclust.boxplot.filename_new`

permute_test *Permutation Test for Compositional Regression Coefficients*

Description

Performs a non-parametric permutation test to calculate empirical p-values for the coefficients from a `CompReg` model.

Usage

```
permute_test(X_screen, y, rep, level)
```

Arguments

`X_screen`	A numeric matrix of log-transformed compositional predictors.
`y`	A numeric vector for the response variable.
`rep`	The number of permutation replicates to perform.
`level`	The significance level (alpha) for flagging results.

Details

This function assesses the statistical significance of each compositional predictor without relying on the assumption of normally distributed errors. It works by:

1. Calculating the observed coefficients (`beta_hat`) and t-statistics (`t_ref`) from the original data.
2. Creating a null distribution by repeatedly (`rep` times) shuffling the response variable `y`. This shuffling breaks the true relationship between predictors and response.
3. For each shuffled `y`, a new `CompReg` model is fitted, and its coefficients and t-statistics are stored.
4. The empirical p-value for each predictor is then calculated as the proportion of times the absolute value of the permuted statistic was greater than or equal to the absolute value of the observed statistic. A small p-value indicates that the observed relationship is unlikely to have occurred by chance.

Value

A list containing data frames with calculated p-values and significance flags, based on both the beta coefficients and the t-statistics.

`png.CompReg` *Run Compositional Regression Analysis*

Description

A high-level wrapper function to perform compositional regression for one or more dependent variables, with options for parametric or non-parametric inference.

Usage

```
png.CompReg(X0, Y, type = c("bootstrap", "parametric"))
```

Arguments

`X0`	A numeric matrix of raw compositional predictors (proportions).
`Y`	A numeric matrix or data frame where each column is a dependent variable.
`type`	A string specifying the inference method: `"parametric"` or `"bootstrap"`.

Details

This is the main driver function for the compositional regression analysis. It iterates through each dependent variable (each column in `Y`) and performs a full analysis. The process for each DV is:

1. Handle missing values.
2. Apply a log-transformation to the compositional predictors `X0`.
3. Fit the core `CompReg` model.
4. Perform statistical inference based on the `type` parameter:
   - `type = "parametric"`: Calculates standard errors, confidence intervals, and p-values based on the t-distribution, assuming model errors are normally distributed.
   - `type = "bootstrap"`: Calls `bootstrap_test` and `permute_test` to derive confidence intervals and p-values from resampling, which does not rely on distributional assumptions.

Value

A list of lists. Each top-level element corresponds to a dependent variable in `Y` and contains the full regression result (estimates, CIs, p-values, etc.).

`reassign.enrichment` *Reassign Members of Non-Significant Clusters*

Description

Reassigns data points from non-significant clusters to the nearest significant cluster based on Euclidean distance.

Usage

```
reassign.enrichment(Xnew, class, mod, wh.class)
```

Arguments

`Xnew`	The numeric data matrix of observations to be reassigned. Should only contain observations from non-significant clusters.
`class`	A vector of original cluster assignments for all observations.
`mod`	The fitted model object (e.g., from `mclust::Mclust`), which must contain the cluster parameters, specifically `mod$parameters$mean`.
`wh.class`	A vector containing the labels of the clusters considered "significant".

Details

This function is a post-processing step for clustering. After identifying a set of "significant" or robust clusters (e.g., via enrichment analysis), this function handles the observations that fall into non-significant clusters.

The reassignment is based on a nearest-centroid rule. For each data point in a non-significant cluster, the function calculates the L2-norm (Euclidean distance) to the centroids (mean vectors) of all *significant* clusters. The point is then reassigned to the cluster with the minimum distance. This method provides a principled way to consolidate a clustering solution by merging less stable or sparsely populated clusters into more robust, well-defined ones, thereby simplifying the final interpretation.

If all initial classes are already within the significant set (`wh.class`), the function returns the original class assignments without change.

Value

A new vector of class assignments of the same length as the original `class` vector, where members of non-significant clusters have been reassigned.

`Xclust.boxplot.filename` *Generate and Save Boxplots of Factor Scores by Cluster*

Description

Creates a faceted boxplot to visualize the distribution of personality factor scores for each identified cluster and saves it to a PDF file.

Usage

```
Xclust.boxplot.filename(Xclust, wh.best, filename)
```

Arguments

`Xclust`	A data frame with factor scores and a 'class' column for cluster assignments.
`wh.best`	A numeric vector of length 2, containing the optimal number of clusters (`wh.best[1]`) and the corresponding random seed (`wh.best[2]`).
`filename`	A base string for the output PDF file name.

Details

This function serves to interpret the results of a clustering analysis (e.g., GMM). It takes a data frame `Xclust` where each row is an observation, columns represent different factor scores (e.g., O, C, E, A, N), and a 'class' column indicates cluster membership.

The data is transformed into a long format, and `ggpubr::ggboxplot` is used to create boxplots of scores for each factor. The plots are faceted by cluster (`facet.by="class"`), allowing for a direct visual comparison of the personality profiles of the different clusters. The resulting plot is saved as a PDF file, with a filename indicating the seed and number of clusters used.

Value

This function does not return a value. It saves a plot to a PDF file.

Index