Package ‘ComBatFamQC’

March 17, 2025

Title Comprehensive Batch Effect Diagnostics and Harmonization

Version 1.0.5

Description Provides a comprehensive framework for batch effect diagnostics, harmonization, and post-harmonization downstream analysis. Features include interactive visualization tools, robust statistical tests, and a range of harmonization techniques. Additionally, 'ComBatFamQC' enables the creation of life-span age trend plots with estimated age-adjusted centiles and facilitates the generation of covariate-corrected residuals for analytical purposes. Methods for harmonization are based on approaches described in Johnson et al., (2007) <doi:10.1093/biostatistics/kxj037>, Beer et al., (2020) <doi:10.1016/j.neuroimage.2020.117129>, Pomponio et al., (2020) <doi:10.1016/j.neuroimage.2019.116450>, and Chen et al., (2021) <doi:10.1002/hbm.25688>.

License MIT + file LICENSE

URL https://github.com/Zheng206/ComBatFamQC,
https://zheng206.github.io/ComBatQC-Web/

BugReports https://github.com/Zheng206/ComBatFamQC/issues

Encoding UTF-8

RoxygenNote 7.3.2

Imports tidyr, dplyr, magrittr, ggplot2, DT, shiny, car, broom, pbkrtest, parallel, Rtsne, MDMR, gamlss, lme4, mgcv, bslib, shinydashboard, methods, gamlss.dist, invgamma, openxlsx

Depends R (>= 4.1.0)

LazyData true

Suggests knitr, rmarkdown, devtools, testthat (>= 3.0.0), remotes, plotly, quarto, spelling,

VignetteBuilder knitr

Config/testthat/edition 3

Language en-US

NeedsCompilation no

Author Zheng Ren [aut, cre, cph] (<https://orcid.org/0009-0006-2430-4625>),
Andrew Chen [aut, cph] (<https://orcid.org/0000-0002-5027-6422>),
Elizabeth Horwath [ctb] (<https://orcid.org/0009-0009-1224-6984>)

Maintainer Zheng Ren <zren1422@gmail.com>

Repository CRAN

Date/Publication 2025-03-17 07:30:19 UTC

Contents

.biweight_midvar *Biweight Midvariance Calculation*

Description

Compute a robust estimate of midvariance using the biweight method, which reduces the influence of outliers by applying a weighting function to the data based on their deviation from a central value.

Usage

```
.biweight_midvar(data, center = NULL, norm.unbiased = TRUE)
```

Arguments

`data` A numeric vector containing the data points for which the biweight midvariance is to be calculated.

`center` An optional parameter specifying the central location of the data. If not provided, the function defaults to using the median of the data.

`norm.unbiased` A logical parameter (default: `TRUE`) indicating whether to use a normalization constant for unbiased estimation. When `TRUE`, the constant is adjusted to 9 divided by the quantile function of 0.75 from the standard normal distribution.

Value

A numeric value representing the robust biweight midvariance estimate.

Examples

```
data <- c(1, 2, 3, 4, 100)
biweight_var <- .biweight_midvar(data)
print(biweight_var)
```

`adni` *Harmonization Data*

Description

This data set is a sample data set from ADNI study.

Usage

```
data(adni)
```

Format

`"adni"` is a data frame with 2515 cases (rows) and 104 variables (columns). 62 features are included for harmonization purpose.

Source

ADNI

Examples

```
data(adni)
head(adni)
```

age_df *Age Trajectory Data*

Description

This data set is used to observe life span age trend of brain structures.

Usage

```
data(age_df)
```

Format

"age_df" is a simulated data frame with 712 cases (rows) and 56 variables (columns). 51 rois are included.

Examples

```
data(age_df)
head(age_df)
```

age_list_gen *Age Trend Estimates Generation*

Description

A GAMLSS model using a Normal distribution was fitted separately to rois of interest, to establish normative reference ranges for the volume of a specific ROI as a function of age, adjusting for sex and intracranial volume.

Usage

```
age_list_gen(
  sub_df,
  lq = 0.25,
  hq = 0.75,
  mu = "smooth",
  sigma = "smooth",
  nu = "default",
  tau = "default"
)
```

Arguments

sub_df A four-column dataset that contains age, sex, intracranial volume (ICV) and roi volume related information. The columns for age, sex, and ICV must be strictly named "age", "sex", and "icv".

lq The lower bound quantile. eg: 0.25, 0.05

hq The upper bound quantile. eg: 0.75, 0.95

mu An indicator of whether to smooth age variable, include it as a linear term or only include the intercept in the mu formula. "smooth": y ~ pb(age), "linear": y ~ age, "default": y ~ 1.

sigma An indicator of whether to smooth age variable, include it as a linear term or only include the intercept in the sigma formula. "smooth": ~ pb(age), "linear": ~ age, "default": ~ 1.

nu An indicator of whether to smooth age variable, include it as a linear term or only include the intercept in the nu formula. "smooth": ~ pb(age), "linear": ~ age, "default": ~ 1.

tau An indicator of whether to smooth age variable, include it as a linear term or only include the intercept in the tau formula. "smooth": ~ pb(age), "linear": ~ age, "default": ~ 1.

Value

age_list_gen returns a list containing the following components:

true_df a dataframe contains the true age and ROI volume information

predicted_df_sex
a dataframe contains the estimated age trend adjusting sex and icv

model the fitted GAMLSS model

Examples

```
sub_df <- age_df[,c("Volume_1", "age", "sex", "ICV_baseline")] |> na.omit()
colnames(sub_df) <- c("Volume_1", "age", "sex", "icv")
age_list_gen(sub_df = sub_df)
```

age_save *Export Brain ROI Age Trends*

Description

Save all brain age trends into a single Excel file.

Usage

```
age_save(path, age_list)
```

Arguments

path	The path to save the excel file.
age_list	A list containing all ROIs' true volumes, age trend estimates, and the fitted GAMLSS model.

Value

This function does not return a value. It saves the data to the specified file.

Examples

```
if(interactive()){
sub_df <- age_df[,c("Volume_1", "age", "sex", "ICV_baseline")] |> na.omit()
colnames(sub_df) <- c("Volume_1", "age", "sex", "icv")
age_list <- list("Volume_1" = age_list_gen(sub_df = sub_df))

temp_dir <- tempfile()
dir.create(temp_dir)
age_save(temp_dir, age_list)
message("Age trend table saved to: ", temp_dir)
unlink(temp_dir, recursive = TRUE)
}
```

---

age_shiny *Lifespan Age Trends*

Description

Provide estimated lifespan age trends of neuroimaging-derived brain structures through shiny app.

Usage

```
age_shiny(age_list, features, quantile_type, use_plotly = TRUE)
```

Arguments

age_list	A list containing all ROIs' true volumes, age trend estimates, and the fitted GAMLSS model.
features	A vector of roi names.
quantile_type	A vector of quantile types (e.g., c("quantile_25", "median", "quantile_75"))
use_plotly	A boolean variable that indicates whether to display the age plot using the plotly package.

Details

When this function is called, it starts a Shiny application in the user's default web browser. Execution is blocked until the app is closed.

Value

This function does not return a value. It launches a Shiny app.

Examples

```
sub_df <- age_df[,c("Volume_1", "age", "sex", "ICV_baseline")] |> na.omit()
colnames(sub_df) <- c("Volume_1", "age", "sex", "icv")
age_list <- list("Volume_1" = age_list_gen(sub_df = sub_df))
quantile_type <- c("quantile_25", "median", "quantile_75")
if(interactive()){
  age_shiny(age_list = age_list, features = "Volume_1", quantile_type = quantile_type)
}
```

age_table_gen *Generate Age Trend Summary Table*

Description

This function generates a summary table of age trend predictions for a specified quantile and demographic group. The table includes the predicted volume values, percentage changes between age intervals, and other details for either females, males, or a comparison between both genders.

Usage

```
age_table_gen(result, q = "median", s = "F")
```

Arguments

`result`	A data object containing prediction results and metadata.
`q`	A string specifying the quantile of interest (e.g., `"quantile_50"` for the median).
`s`	A string indicating the demographic group for which the summary table is generated: • `"F"` for female • `"M"` for male • `"F vs M"` for comparison between females and males

Details

The function processes the input data to filter predictions based on the specified quantile and demographic group. It calculates percentage changes in predicted volume values for easier interpretation of trends. For gender comparisons (`"F vs M"`), it generates side-by-side columns for females and males.

The output table is formatted using the `DT` package with additional features, such as CSV and Excel export options.

Value

A datatable object (from the DT package) containing the age trend summary table with the following columns:

- Age: The age values at regular intervals (rounded to the nearest 10).
- Percentile.Volume: The predicted volume values for the specified quantile (only for females or males).
- PercentageChange (%): The percentage change in volume between consecutive age intervals (only for females or males).
- Percentile.Volume_F: The predicted volume values for females (when comparing genders).
- Percentile.Volume_M: The predicted volume values for males (when comparing genders).
- PercentageChange_F (%): The percentage change for females (when comparing genders).
- PercentageChange_M (%): The percentage change for males (when comparing genders).

Examples

```
sub_df <- age_df[,c("Volume_1", "age", "sex", "ICV_baseline")] |> na.omit()
colnames(sub_df) <- c("Volume_1", "age", "sex", "icv")
age_list <- list("Volume_1" = age_list_gen(sub_df = sub_df))
result <- age_list[[1]]
if(interactive()){
 age_table_gen(result, q = "median", s = "F")
}
```

age_trend_plot *Generate Age Trend Plot*

Description

This function creates an age trend plot for a specified feature and demographic group based on GAMLSS model predictions. The function supports both static plots using ggplot2 and interactive plots using plotly.

Usage

```
age_trend_plot(
  age_list,
  f,
  s = "none",
  q = "median",
  cus_list = NULL,
  use_plotly = TRUE
)
```

Arguments

`age_list` A list containing all ROIs' true volumes, age trend estimates, and the fitted GAMLSS model.

`f` A string specifying the feature of interest within the `age_list`.

`s` A string indicating the demographic group for visualization: `"F"` for female, `"M"` for male, `"F vs M"` for gender comparison, or `"none"` for base plot (true data) visualization.

`q` A string specifying the quantile type (e.g., `"quantile_75"`, `"median"`, or `"customization"`).

`cus_list` A list object containing customized quantile predictions generated by the `cus_result_gen` function.

`use_plotly` A boolean indicating whether to create an interactive plot using `plotly` (default: `TRUE`).

Details

The function overlays true data points with predicted quantile trends for the specified feature and demographic group. It supports customization for quantile visualization and uses precomputed results from the `cus_result_gen` function for enhanced flexibility.

Value

A plot object:

- If `use_plotly` = `TRUE`, returns a `plotly` interactive plot.
- If `use_plotly` = `FALSE`, returns a `ggplot2` static plot.

Examples

```
sub_df <- age_df[,c("Volume_1", "age", "sex", "ICV_baseline")] |> na.omit()
colnames(sub_df) <- c("Volume_1", "age", "sex", "icv")
age_list <- list("Volume_1" = age_list_gen(sub_df = sub_df))
customized_results <- cus_result_gen(age_list, customized_q = 0.75, f = "Volume_1")

if(interactive()){
 age_trend_plot(
   age_list = age_list,
   f = "Volume_1",
   s = "F",
   q = "customization",
   cus_list = customized_results,
   use_plotly = TRUE
 )
}
```

combat_harm *ComBatFamily Harmonization*

Description

Conduct harmonization using four types of methods: 1) Original ComBat, 2) Longitudinal ComBat, 3) ComBat-GAM, and 4) CovBat.

Usage

```
combat_harm(
  eb_check = FALSE,
  result = NULL,
  features = NULL,
  batch = NULL,
  covariates = NULL,
  df = NULL,
  type = "lm",
  random = NULL,
  smooth = NULL,
  interaction = NULL,
  smooth_int_type = NULL,
  family = "comfam",
  eb = TRUE,
  ref.batch = NULL,
  predict = FALSE,
  object = NULL,
  reference = NULL,
  out_ref_include = TRUE,
  ...
)
```

Arguments

`eb_check`	A boolean variable indicating whether the user wants to run the EB assumption test before harmonization.
`result`	A list derived from `visual_prep()` that contains dataset and batch effect diagnostic information for Shiny visualization. Can be skipped if `features`, `batch`, `covariates` and `df` are provided.
`features`	The name of the features to be harmonized. This can be skipped if `result` is provided.
`batch`	The name of the batch variable. Can be skipped if `result` is provided.
`covariates`	The names of covariates supplied to `model`. This can be be skipped if `result` is provided.
`df`	Dataset to be harmonized. This can be be skipped if `result` is provided.

type The name of a regression model to be used: "lmer", "lm", "gam".

random The variable name of a random effect in linear mixed effect model.

smooth The name of the covariates that require a smooth function.

interaction Expression of interaction terms supplied to model (eg: "age,diagnosis").

smooth_int_type
A vector that indicates the types of interaction in gam models. By default, smooth_int_type is set to be NULL, "linear" represents linear interaction terms. "categorical-continuous", "factor-smooth" both represent categorical-continuous interactions ("factor-smooth" includes categorical variable as part of the smooth), "tensor" represents interactions with different scales, and "smooth-smooth" represents interaction between smoothed variables.

family The type of combat family to use, comfam or covfam.

eb If TRUE, uses ComBat model with empirical Bayes for mean and variance harmonization

ref.batch The name of the reference batch.

predict A boolean variable indicating whether to run ComBat from scratch or apply existing model to new dataset (currently only work for original ComBat and ComBat-GAM).

object Existing ComBat model.

reference Dataset to be considered as the reference group.

out_ref_include
A boolean variable indicating whether the reference data should be included in the harmonized data output.

... Additional arguments to comfam or covfam models.

Value

If the eb_check is set to be FALSE, then combat_harm returns a list containing the following components:

com_family ComBat family to be considered: comfam, covfam

harmonized_df Harmonized dataset

combat.object Saved ComBat model and relevant information, such as the batch variable name and whether the EB method is used

If eb_check is set to be TRUE, then combat_harm will return a dataframe with the EB assumption test result.

Examples

```
combat_harm(features = colnames(adni)[43:53], batch = "manufac",
covariates = c("AGE", "SEX", "DIAGNOSIS"), df = head(adni,100), type = "lm")
```

combat_plot_gen *Generate Diagnostic Plots for Batch Effect Analysis*

Description

This function generates a variety of diagnostic plots for analyzing batch effects and their relationships with features and covariates. Depending on the specified plot type, it can create density plots, box plots, residual plots, PCA plots, T-SNE plots, and empirical Bayes diagnostic plots.

Usage

```
combat_plot_gen(
  result,
  f = NULL,
  batch_control = "No",
  batch_level = NULL,
  plot_name,
  c = NULL,
  smooth_method = "lm",
  alpha = 0.2,
  char_plot_type = "boxplot",
  text_status = "No",
  color = "No",
  label = "No",
  angle = 0,
  PC1 = NULL,
  PC2 = NULL,
  eb = TRUE,
  eb_df = NULL
)
```

Arguments

result	A list derived from `visual_prep()` that contains datasets and statistical test results for Shiny visualization.
f	A string specifying the feature of interest for visualization.
batch_control	A string indicating whether to include batch-specific controls. Defaults to `"No"`.
batch_level	A vector specifying the batch levels to include in the plot. Used only when `batch_control` is not `"No"`.
plot_name	A string specifying the type of plot to generate. Options include `"batch_density"`, `"cov_feature"`, `"batch_summary"`, `"cov_distribution"`, `"resid_add"`, `"resid_mul"`, `"pca"`, `"tsne"`, `"eb_location"`, and `"eb_scale"`.
c	A string specifying the covariate of interest for `"cov_feature"` or `"cov_distribution"` plots.
smooth_method	A string specifying the smoothing method for trend lines. Defaults to `"lm"` (linear model).

`alpha` A numeric value between 0 and 1 controlling the transparency of trend lines. Defaults to `0.2`.

`char_plot_type` A string specifying the type of plot for categorical covariates. Options include `"boxplot"`, `"boxplot with points"`, and `"density plot"`. Defaults to `"boxplot"`.

`text_status` A string indicating whether to display text annotations in the plot. Defaults to `"No"`.

`color` A string indicating whether to use color coding in plots. Defaults to `"No"`.

`label` A string indicating whether to include axis labels in the plot. Defaults to `"No"`.

`angle` A numeric value specifying the angle of x-axis labels. Defaults to `0`.

`PC1` A string specifying the first principal component for PCA plots.

`PC2` A string specifying the second principal component for PCA plots.

`eb` A logical value indicating whether to include empirical Bayes prior information in the plot. Defaults to `TRUE`.

`eb_df` A data frame containing empirical Bayes information for generating `eb_location` and `eb_scale` plots.

Details

The function dynamically generates plots based on the `plot_name` parameter:

- `"batch_density"`: Density plots of features by batch levels.
- `"cov_feature"`: Covariate vs. feature plots with optional batch adjustments.
- `"batch_summary"`: Bar plots summarizing batch-level distributions.
- `"cov_distribution"`: Covariate distributions stratified by batch.
- `"resid_add"`: Additive residual box plots.
- `"resid_mul"`: Multiplicative residual box plots.
- `"pca"`: Principal Component Analysis (PCA) plots.
- `"tsne"`: T-SNE plots for dimensionality reduction.
- `"eb_location"`: Empirical Bayes location parameter density plots.
- `"eb_scale"`: Empirical Bayes scale parameter density plots.

Value

A ggplot object representing the specified diagnostic plot.

Examples

```
if(interactive()){
 result <- visual_prep(type = "lm", features = "thickness.left.cuneus",
 batch = "manufac", covariates = "AGE", df = adni[1:100, ], mdmr = FALSE, cores = 1)
 combat_plot_gen(result, f = "thickness.left.cuneus", plot_name = "batch_density")
 combat_plot_gen(result, f = "thickness.left.cuneus", c = "AGE", plot_name = "cov_feature")
}
```

combat_table_gen *Generate Diagnostic Tables for Batch Effect Analysis*

Description

This function generates a variety of tables to summarize data and results from batch effect analyses. Depending on the specified table name, it can create data overview tables, exploratory analysis summaries, statistical test results, PCA summaries, and covariate distributions.

Usage

```
combat_table_gen(
  result,
  table_name,
  f = NULL,
  c = NULL,
  PC1 = NULL,
  PC2 = NULL
)
```

Arguments

`result` A list derived from `visual_prep()` that contains datasets and statistical test results for Shiny visualization.

`table_name` A string specifying the type of table to generate. Options include:

- `"data_overview"`: Overview of the dataset, including covariates, features, and batch information.
- `"exploratory_analysis"`: Summary of the selected feature and covariates.
- `"summary_df"`: Summary of batch-level distributions, including batch removal status.
- `"cov_table"`: Covariate summary table, displaying distributions for numeric or categorical covariates.
- `"pc_variance"`: Variance explained by specified principal components.
- `"mdmr"`: Results from the Multivariate Distance Matrix Regression (MDMR) analysis.
- `"kenward_roger"`: Results from Kenward-Roger tests.
- `"anova"`: Results from ANOVA tests.
- `"kruskal_wallis"`: Results from Kruskal-Wallis tests.
- `"fligner_killeen"`: Results from Fligner-Killeen tests.
- `"levenes"`: Results from Levene’s tests.
- `"bartletts"`: Results from Bartlett’s tests.

`f` A string specifying the feature of interest for tables requiring a specific feature. Default is `NULL`.

c	A string specifying the covariate of interest for tables requiring a specific covariate. Default is NULL.
PC1	A string specifying the first principal component for PCA variance tables. Default is NULL.
PC2	A string specifying the second principal component for PCA variance tables. Default is NULL.

Details

The function dynamically generates tables based on the `table_name` parameter.

Value

A `DT::datatable` object containing the requested table.

Examples

```
if(interactive()){
 result <- visual_prep(type = "lm", features = "thickness.left.cuneus",
 batch = "manufac", covariates = "AGE", df = adni[1:100, ], mdmr = FALSE, cores = 1)
 combat_table_gen(result, table_name = "cov_table", c = "AGE")
 combat_table_gen(result, table_name = "pc_variance", PC1 = "PC1", PC2 = "PC2")
 }
```

---

comfam *ComBat Family Harmonization*

Description

Implementation of the ComBat Family of harmonization methods allowing for flexible covariate modeling and alternative estimators for site effect adjustment. Support for modeling of both location and scale via GAMLSS and longitudinal harmonization via mixed effects models.

Usage

```
comfam(
  data,
  bat,
  covar = NULL,
  model = lm,
  formula = NULL,
  eb = TRUE,
  robust.LS = FALSE,
  ref.batch = NULL,
  ...
)
```

Arguments

data	*n x p* data frame or matrix of observations where *p* is the number of features and *n* is the number of subjects.
bat	Factor indicating batch (often equivalent to site or scanner)
covar	Data frame or matrix of covariates supplied to `model`
model	Model function. ComBat Family supports any models that take arguments `formula` and `data`, but are limited to models fitting with identity link (e.g. `family = gaussian(link = "identity")`). This includes lm, gam, gamlss, rq, lmer, and more
formula	Formula for `model` starting with `y ~` where `y` represents each feature
eb	If `TRUE`, uses ComBat model with empirical Bayes for mean and variance harmonization
robust.LS	If `TRUE`, uses robust location and scale estimators for error variance and site effect parameters. Currently uses median and biweight midvariance
ref.batch	Reference batch, must take value in `levels(bat)`
...	Additional arguments to `model`

Value

`comfam` returns a list containing the following components:

dat.combat	Harmonized data as a matrix with same dimensions as `data`
batch.info	Batch information, including reference batch if specified
fits	List of model fits from regression step, outputs of `model` for each feature
estimates	List of estimates from standardization and batch effect correction

See Also

predict.comfam for applying ComBat parameters for harmonization of new observations

Examples

```
comfam(iris[,1:2], iris$Species)
comfam(iris[,1:2], iris$Species, iris[3:4], lm, y ~ Petal.Length + Petal.Width)
```

comfam_shiny *Batch Effect Interactive Visualization*

Description

Provides an interactive visualization of batch or site effects using a Shiny application.

Usage

```
comfam_shiny(result, after = FALSE)
```

Arguments

result	A list derived from visual_prep() that contains datasets and statistical test results for Shiny visualization.
after	A boolean variable indicating whether the batch effect diagnostic occurs before or after harmonization (default: FALSE).

Details

When this function is called, it starts a Shiny application in the user's default web browser. Execution is blocked until the app is closed.

Value

This function does not return a value. It launches a Shiny app.

Examples

```
result_lm <- visual_prep(type = "lm", features = colnames(adni)[43:53],
batch = "manufac", covariates = c("AGE", "SEX", "DIAGNOSIS"),
df = head(adni, 500), cores = 1)
if (interactive()) {
  comfam_shiny(result = result_lm)
}
```

covfam *CovBat Family Harmonization*

Description

Implementation of the CovBat Family of harmonization methods allowing for removal of multivariate batch effects, flexible covariate modeling and alternative estimators for site effect adjustment. Support for modeling of both location and scale via GAMLSS. Additional support for modeling of covariate effects in score location and scale.

Usage

```
covfam(
  data,
  bat,
  covar = NULL,
  model = lm,
  formula = NULL,
  score.model = NULL,
  score.args = NULL,
  eb = TRUE,
  robust.LS = FALSE,
  ref.batch = NULL,
  percent.var = 0.95,
  n.pc = NULL,
  std.var = TRUE,
  ...
)
```

Arguments

`data` *n x p* data frame or matrix of observations where *p* is the number of features and *n* is the number of subjects.

`bat` Factor indicating batch (often equivalent to site or scanner)

`covar` Data frame or matrix of covariates supplied to `model`

`model` Model function. ComBat Family supports any models that take arguments `formula` and `data`, but are limited to models fitting with identity link (e.g. `family = gaussian(link = "identity")`). This includes lm, gam, gamlss, rq, lmer, and more

`formula` Formula for `model` starting with `y ~` where `y` represents each feature

`score.model` Model for scores, defaults to NULL for fitting basic location and scale model without covariates on the scores

`score.args` List of arguments for score model

`eb` If `TRUE`, uses ComBat model with empirical Bayes for mean and variance harmonization.

`robust.LS` If `TRUE`, uses robust location and scale estimators for error variance and site effect parameters. Uses median and biweight midvariance

`ref.batch` Reference batch, must take value in `levels(bat)`

`percent.var` Numeric. The number of harmonized principal component scores is selected to explain this proportion of the variance

`n.pc` Optional numeric. If specified, this number of principal component scores is harmonized. Overrides `percent.var`

`std.var` If `TRUE`, scales variances to be equal to 1 before PCA.

`...` Additional arguments to `model`

Value

`covfam` returns a list containing the following components:

`dat.covbat` Harmonized data as a matrix with same dimensions as `data`

`batch.info` Batch information, including reference batch if specified

`combat.out` List output of `comfam` from the ComBat step

`pc.output` Output of `prcomp` from PCA step

`n.pc` Numeric, number of PCs harmonized

`scores.com` List output of `comfam` from the CovBat step

Examples

```
covfam(iris[,1:2], iris$Species)
covfam(iris[,1:2], iris$Species, iris[3:4], lm, y ~ Petal.Length + Petal.Width)
```

customize_percentile *Generate Predicted Quantiles for Age Trends*

Description

This function computes predicted quantiles for a specified feature and demographic group based on a GAMLSS model. The function interpolates predictions over a range of ages while accounting for fixed covariates.

Usage

```
customize_percentile(age_list, feature, q = 0.75, s = "F")
```

Arguments

age_list	A list containing all ROIs' true volumes, age trend estimates, and the fitted GAMLSS model.
feature	A string specifying the feature of interest within the age_list.
q	A numeric value between 0 and 1 representing the quantile to predict (e.g., 0.5 for the median).
s	A string indicating the gender of the group for which the predictions are generated (e.g., "F" for female, "M" for male).

Details

This function uses a GAMLSS model to generate predictions for a specified quantile and demographic group. The predictions are computed over a sequence of ages (age_test) that spans the observed age range in the data. The function adjusts for fixed covariates such as icv by using their mean values from the input data.

Value

A data frame containing columns for age, quantile type, prediction, and sex.

Examples

```
sub_df <- age_df[,c("Volume_1", "age", "sex", "ICV_baseline")] |> na.omit()
colnames(sub_df) <- c("Volume_1", "age", "sex", "icv")
age_list <- list("Volume_1" = age_list_gen(sub_df = sub_df))
customize_percentile(age_list, feature = "Volume_1", q = 0.5, s = "F")
```

cus_result_gen *Generate Customized Predicted Quantiles List*

Description

This function computes customized predicted quantiles for a specified feature across both female and male groups using a GAMLSS model. The resulting list object is structured for visualization purposes.

Usage

```
cus_result_gen(age_list, customized_q = 0.75, f)
```

Arguments

`age_list`	A list containing all ROIs' true volumes, age trend estimates, and the fitted GAMLSS model.
`customized_q`	A numeric value between 0 and 1 representing the quantile to predict (e.g., `0.75` for the 75th percentile).
`f`	A string specifying the feature of interest within the `age_list`.

Details

This function utilizes the GAMLSS model to compute predictions for the specified quantile across demographic groups. The results include predictions for both the specified quantile and the median, enabling enhanced visualization.

Value

A list containing the customized quantile value used for predictions and a data frame with columns for age, quantile type, prediction, and sex.

Examples

```
sub_df <- age_df[,c("Volume_1", "age", "sex", "ICV_baseline")] |> na.omit()
colnames(sub_df) <- c("Volume_1", "age", "sex", "icv")
age_list <- list("Volume_1" = age_list_gen(sub_df = sub_df))
cus_result_gen(age_list, customized_q = 0.75, f = "Volume_1")
```

data_prep *Data Preparation*

Description

Prepares the dataset for effective use in batch effect diagnostics, harmonization, and post-harmonization downstream analysis processes within the ComBatFamQC package.

Usage

```
data_prep(
  stage = "harmonization",
  result = NULL,
  features = NULL,
  batch = NULL,
  covariates = NULL,
  df = NULL,
  type = "lm",
  random = NULL,
  smooth = NULL,
  interaction = NULL,
  smooth_int_type = NULL,
  predict = FALSE,
  object = NULL
)
```

Arguments

`stage` Specifies the stage of analysis for which the data preparation is intended: harmonization or residual.

`result` A list derived from `visual_prep()` that contains dataset and batch effect diagnostic information for Shiny visualization. Can be skipped if `features`, `batch`, `covariates` and `df` are provided.

`features` The name of the features to be harmonized. This can be skipped if `result` is provided.

`batch` The name of the batch variable. Can be skipped if `result` is provided.

`covariates` The names of covariates supplied to `model`. This can be be skipped if `result` is provided.

`df` The dataset to be harmonized. This can be be skipped if `result` is provided.

`type` The name of a regression model to be used in batch effect diagnostics, harmonization, and the post-harmonization stage: "lmer", "lm", "gam".

`random` The variable name of a random effect in linear mixed effect model.

`smooth` The name of the covariates that require a smooth function.

`interaction` Expression of interaction terms supplied to `model` (eg: "age,diagnosis").

smooth_int_type
: A vector that indicates the types of interaction in gam models. By default, smooth_int_type is set to be NULL, "linear" represents linear interaction terms. "categorical-continuous", "factor-smooth" both represent categorical-continuous interactions ("factor-smooth" includes categorical variable as part of the smooth), "tensor" represents interactions with different scales, and "smooth-smooth" represents interaction between smoothed variables.

predict
: A boolean variable indicating whether to run ComBat from scratch or apply existing model to new dataset (currently only work for "original ComBat" and "ComBat-GAM").

object
: Existing ComBat model.

Value

data_prep returns a list containing the processed data and parameter-related information for batch effect diagnostics, harmonization, and post-harmonization downstream analysis.

Examples

```
data_prep(stage = "harmonization", result = NULL, features = colnames(adni)[43:53],
batch = "manufac", covariates = "AGE", df = head(adni, 100), type = "lm", random = NULL,
smooth = NULL, interaction = NULL, smooth_int_type = NULL, predict = FALSE, object = NULL)
```

diag_save *Export Batch Effect Diagnosis Results*

Description

Save all the batch effect diagnosis results in a single Excel file or a Quarto report.

Usage

```
diag_save(path, result, use_quarto = TRUE)
```

Arguments

path
: The path to save the result.

result
: A list derived from visual_prep() that contains datasets and statistical test results.

use_quarto
: A boolean variable indicating whether to generate a Quarto report.

Value

This function does not return a value. It saves the data to the specified file.

Examples

```
if(interactive()){
  result <- visual_prep(type = "lm", features = "thickness.left.cuneus",
  batch = "manufac", covariates = "AGE", df = adni[1:100, ], mdmr = FALSE, cores = 1)
  temp_dir <- tempfile()
  dir.create(temp_dir)
  diag_save(temp_dir, result, quarto = FALSE)
  message("Diagnostics saved to: ", temp_dir)
  unlink(temp_dir, recursive = TRUE)  # Clean up the temporary directory
}
```

eb_check *EB Assumption Check*

Description

Generate the empirical and prior distribution of both the location parameter gamma and the scale parameter delta.

Usage

```
eb_check(
  data,
  bat,
  covar = NULL,
  model = lm,
  formula = NULL,
  robust.LS = FALSE,
  ref.batch = NULL,
  ...
)
```

Arguments

data	*n x p* data frame or matrix of observations where *p* is the number of features and *n* is the number of subjects.
bat	Factor indicating batch (often equivalent to site or scanner).
covar	Data frame or matrix of covariates supplied to model.
model	The model function that ComBat Family supports: lm, lmer, gam.
formula	Formula for model starting with y ~ where y represents each feature.
robust.LS	If TRUE, uses robust location and scale estimators for error variance and site effect parameters. Currently uses median and biweight midvariance.
ref.batch	Reference batch, must take value in levels(bat).
...	Additional arguments to model.

Value

eb_check returns a dataframe containing the empirical and prior distribution of both the location parameter (gamma) and the scale parameter (delta).

Examples

```
eb_check(data = adni[1:500,43:53], bat = as.factor(adni$manufac[1:500]),
covar = adni[1:500, c("AGE", "SEX")], model = lm, formula = y ~ AGE + SEX)
```

form_gen *ComBatFamily Model Formula Generations*

Description

Generate appropriate formula for ComBatFamily models

Usage

```
form_gen(x, c = NULL, i = NULL, random = NULL, smooth = NULL)
```

Arguments

x	A model function name that is used or to be used in the ComBatFamily Package (eg: "lmer", "lm", "gam").
c	Data frame or matrix of covariates supplied to model.
i	Expression of interaction terms supplied to model. (eg: "age:diagnosis").
random	Variable name of a random effect in linear mixed effect model.
smooth	Variable name that requires a smooth function.

Value

A string of formula

Examples

```
covariates <- adni[, c("AGE", "SEX")]
form_gen(x = "lm", c = covariates)
```

getQuantileRefactored *Compute Quantile Functions for a Predictor in a GAMLSS Model*

Description

This function computes quantile functions for a specified predictor in a GAMLSS model, allowing the evaluation of response quantiles (e.g., 25th, 50th, 75th percentiles) as a function of the predictor. It is a modified version of the `getQuantile` function from the **GAMLSS** package, with improvements to explicitly require the dataset as an argument, avoiding reliance on global or external variables.

Usage

```
getQuantileRefactored(
  obj,
  term,
  quantile,
  data,
  n.points = 100,
  fixed.at = list()
)
```

Arguments

`obj`	A fitted GAMLSS model object.
`term`	A character string specifying the name of the predictor variable for which quantiles are computed.
`quantile`	A numeric vector of probabilities (e.g., `c(0.25, 0.5, 0.75)`) at which to compute the quantiles.
`data`	A data frame containing the dataset used in the model. This must include the predictor specified in `term`.
`n.points`	An integer specifying the number of points at which to evaluate the quantile functions (default: 100).
`fixed.at`	A named list specifying fixed values for other predictors in the dataset. If not provided, categorical predictors are set to their most frequent levels, and numeric predictors to their median values.

Details

This function generates a temporary dataset by varying the specified predictor (`term`) over a sequence of values while holding all other predictors constant (using values specified in `fixed.at`, or derived defaults). It then computes the distribution parameters for the GAMLSS model and calculates the quantiles using the appropriate quantile function for the distribution family.

Value

A list of spline functions, one for each quantile specified in `quantile`. Each spline function can be evaluated at specific predictor values to obtain the corresponding quantile.

Examples

```
if (requireNamespace("gamlss", quietly = TRUE)) {
  library(gamlss)
  sub_df <- data.frame(
    age = seq(1, 20, length.out = 100),
    height = 50 + 2.5 * seq(1, 20, length.out = 100) + rnorm(100, 0, 5)
  )

  mdl <- gamlss(height ~ pb(age), data = sub_df, family = NO())

  quantile_function <- getQuantileRefactored(
    obj = mdl,
    term = "age",
    quantile = c(0.25, 0.5, 0.75),
    data = sub_df
  )
 }else{
 message("The 'gamlss' package is not installed. Please install it to run this example.")
 }
```

`interaction_gen` *Interaction Term Generation*

Description

Generate appropriate interaction terms for regression models.

Usage

```
interaction_gen(
  type = "lm",
  covariates = NULL,
  smooth = NULL,
  interaction = NULL,
  smooth_int_type = NULL
)
```

Arguments

`type`	The type of model to be used for batch effect evaluation or harmonization (eg: "lmer", "lm", "gam").
`covariates`	Name of covariates supplied to `model`.
`smooth`	Variable names that require a smooth function.

`interaction` Expression of interaction terms supplied to `model` (eg: "age,diagnosis").

`smooth_int_type` A vector that indicates the types of interaction in `gam` models. By default, smooth_int_type is set to be NULL, "linear" represents linear interaction terms. "categorical-continuous", "factor-smooth" both represent categorical-continuous interactions ("factor-smooth" includes categorical variable as part of the smooth), "tensor" represents interactions with different scales, and "smooth-smooth" represents interaction between smoothed variables.

Value

`interaction_gen` returns a list containing the following components:

`interaction` A vector of interaction terms to be included

`covariates` Modified covariates after expressing interaction terms

`smooth` Modified smooth terms after expressing interaction terms

Examples

```
interaction_gen(type = "lm", covariates = c("AGE", "SEX", "DIAGNOSIS"),
interaction = "AGE,DIAGNOSIS")

interaction_gen(type = "gam", covariates = c("AGE", "SEX", "DIAGNOSIS"),
smooth = "AGE", smooth_int_type = "linear", interaction = "AGE,DIAGNOSIS")
```

`model_gen` *Model Generations*

Description

Generate appropriate regression models based on the model type and formula

Usage

```
model_gen(
  y,
  type = "lm",
  batch = NULL,
  covariates = NULL,
  interaction = NULL,
  random = NULL,
  smooth = NULL,
  df
)
```

Arguments

`y` Dependent variable in the model.

`type` A model function name that is used or to be used in the ComBatFamily Package (eg: "lmer", "lm", "gam").

`batch` Name of batch variable (often equivalent to site or scanner).

`covariates` Name of covariates supplied to `model`.

`interaction` Expression of interaction terms supplied to `model` (eg: "age:diagnosis").

`random` Variable name of a random effect in linear mixed effect model.

`smooth` Variable name that requires a smooth function.

`df` Dataset to be harmonized.

Value

A regression model object to be used for batch effect diagnostics and the post-harmonization stage.

Examples

```
model_gen(y = "thickness.left.caudal.anterior.cingulate", type = "lm",
batch = "manufac", covariates = c("AGE", "SEX"), df = adni)
```

predict.comfam *Apply Harmonization to New Data*

Description

Using parameters estimated via `comfam`, apply harmonization on new data. `predict.comfam` will estimate new batch adjustments if new batches are specified. For batches with existing estimates, the estimates from `object` are used. Harmonization targets are the same as `object` (e.g. `ref.batch` from `object` if specified).

Usage

```
## S3 method for class 'comfam'
predict(
  object,
  newdata,
  newbat,
  newcovar = NULL,
  robust.LS = FALSE,
  eb = TRUE,
  ...
)
```

Arguments

`object` Object of class `comfam`, typically output of the harmonization function in this package.

`newdata` $n\ x\ p$ data frame or matrix of new observations where p is the number of features and n is the number of subjects. The features must match the original `data` used in `object`

`newbat` Factor indicating new batch (often equivalent to site or scanner)

`newcovar` Data frame or matrix of new covariates supplied to `model`. Must contain all variables specified in the original `formula` used in `object`.

`robust.LS` If `TRUE`, uses robust location and scale estimators for new batch effect estimates Currently uses median and biweight midvariance

`eb` If `TRUE`, uses ComBat model with empirical Bayes for new batches

`...` Additional arguments to `predict` for the class of `model` (e.g. `predict.lm` for ComBat)

Details

Note: The function currently does not support models of class `lmer` (e.g., from lmer).

Value

`predict.comfam` returns a list containing the following components:

`dat.combat` New harmonized data as a matrix with same dimensions as `newdata`

`batch.info` New batch information, including reference batch if specified

`fits` List of model fits from regression step, forwarded from `object`

`estimates` List of estimates from standardization and batch effect correction, including new batches if relevant

Examples

```
com_out <- comfam(iris[1:75,1:2], iris$Species[1:75])

# out-of-sample with new batch
out_pred <- predict(com_out, iris[76:150,1:2], iris$Species[76:150])

# in-sample
in_pred <- predict(com_out, iris[1:25,1:2], iris$Species[1:25])
max(in_pred$dat.combat - com_out$dat.combat[1:25,])
```

`residual_gen` *Post Harmonization Residual Generation*

Description

Extract residuals after harmonization.

Usage

```
residual_gen(
  type = "lm",
  features = NULL,
  covariates = NULL,
  interaction = NULL,
  random = NULL,
  smooth = NULL,
  smooth_int_type = NULL,
  df,
  rm = NULL,
  model = FALSE,
  model_path = NULL,
  cores = detectCores()
)
```

Arguments

`type` A model function name that is to be used (eg: `"lmer"`, `"lm"`, `"gam"`).

`features` The names of the features from which to extract residuals.

`covariates` Name of covariates supplied to `model`.

`interaction` Expression of interaction terms supplied to `model` (eg: `"age,diagnosis"`).

`random` Variable name of a random effect in linear mixed effect model.

`smooth` Variable name that requires a smooth function.

`smooth_int_type`
Indicates the type of interaction in `gam` models. By default, `smooth_int_type` is set to be `"linear"`, representing linear interaction terms. `"categorical-continuous"`, `"factor-smooth"` both represent categorical-continuous interactions (`"factor-smooth"` includes categorical variable as part of the smooth), `"tensor"` represents interactions with different scales, and `"smooth-smooth"` represents interaction between smoothed variables.

`df` Harmonized dataset to extract residuals from.

`rm` variables to remove effects from.

`model` A boolean variable indicating whether an existing model is to be used.

`model_path` path to the existing model.

`cores` number of cores used for parallel computing.

Value

`residual_gen` returns a list containing the following components:

`model` a list of regression models for all rois

`residual` Residual dataframe

Examples

```
features <- colnames(adni)[43:53]
residual_gen(type = "lm", features = features,
covariates = c("AGE", "SEX", "DIAGNOSIS"), df = adni, rm = c("AGE", "SEX"), cores = 1)
```

`visual_prep` *Batch Effect Diagnostic Visualization Preparation*

Description

Prepare relevant datasets and statistical test results for batch/site effect diagnostic visualization.

Usage

```
visual_prep(
  type = "lm",
  features,
  batch,
  covariates = NULL,
  interaction = NULL,
  random = NULL,
  smooth = NULL,
  smooth_int_type = NULL,
  df,
  cores = detectCores(),
  mdmr = TRUE
)
```

Arguments

`type` The name of a regression model to be used in batch effect diagnostics stage: `"lmer"`, `"lm"`, `"gam"`.

`features` The name of the features to be evaluated.

`batch` The name of the batch variable.

`covariates` Name of covariates supplied to `model`.

`interaction` Expression of interaction terms supplied to `model` (eg: `"age,diagnosis"`).

`random` Variable name of a random effect in linear mixed effect model.

`smooth` Variable name that requires a smooth function.

`smooth_int_type` Indicates the type of interaction in `gam` models. By default, `smooth_int_type` is set to be `"linear"`, representing linear interaction terms. `"categorical-continuous"`, `"factor-smooth"` both represent categorical-continuous interactions (`"factor-smooth"` includes categorical variable as part of the smooth), `"tensor"` represents interactions with different scales, and `"smooth-smooth"` represents interaction between smoothed variables.

`df` Dataset to be evaluated.

`cores` number of cores used for parallel computing.

`mdmr` A boolean variable indicating whether to run the MDMR test (default: `TRUE`).

Value

`visual_prep` returns a list containing the following components:

`residual_add_df` Residuals that might contain additive and multiplicative joint batch effects

`residual_ml_df` Residuals that might contain multiplicative batch effect

`pr.feature` PCA results

`pca_summary` A dataframe containing the variance explained by Principal Components (PCs)

`pca_df` A dataframe contains features in the form of PCs

`tsne_df` A dataframe prepared for T-SNE plots

`kr_test_df` A dataframe contains Kenward-Roger(KR) test results

`fk_test_df` A dataframe contains Fligner-Killeen(FK) test results

`mdmr.summary` A dataframe contains MDMR results

`anova_test_df` A dataframe contains ANOVA test results

`kw_test_df` A dataframe contains Kruskal-Wallis test results

`lv_test_df` A dataframe contains Levene's test results

`bl_test_df` A dataframe contains Bartlett's test results

`red` A parameter to highlight significant p-values in result table

`info` A list contains input information like batch, covariates, df etc

Examples

```
visual_prep(type = "lm", features = colnames(adni)[43:53], batch = "manufac",
covariates = c("AGE", "SEX", "DIAGNOSIS"), df = head(adni, 500), cores = 1)
```

Index