Package: PepSetTest (via r-universe)

October 22, 2026

Title Peptide Set Test

Version 1.6.0

Description Peptide Set Test (PepSetTest) is a peptide-centric strategy to infer differentially expressed proteins in LC-MS/MS proteomics data. This test detects coordinated changes in the expression of peptides originating from the same protein and compares these changes against the rest of the peptidome. Compared to traditional aggregation-based approaches, the peptide set test demonstrates improved statistical power, yet controlling the Type I error rate correctly in most cases. This test can be valuable for discovering novel biomarkers and prioritizing drug targets, especially when the direct application of statistical analysis to protein data fails to provide substantial insights.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Imports dplyr, limma, lme4, MASS, matrixStats, reshape2, stats, tibble, SummarizedExperiment, methods

Suggests statmod, BiocStyle, knitr, rmarkdown, tidyr

biocViews DifferentialExpression, Regression, Proteomics, MassSpectrometry

VignetteBuilder knitr

URL https://github.com/JmWangBio/PepSetTest

BugReports https://github.com/JmWangBio/PepSetTest/issues

Config/pak/sysreqs cmake make libicu-dev zlib1g-dev

Repository Bioconductor 3.23

Date/Publication 2026-04-28 13:03:21 UTC

RemoteUrl https://github.com/bioc/PepSetTest

RemoteRef RELEASE_3_23

RemoteSha f4a08198c4318d856c39f6e95e4df0a0c2e853d8

Contents

AggLimmaWorkflow 2
AggPeps 4
CompPepSetTest 6
CompPepSetTestWorkflow 7
EnframeContrastsRes 10
EstimInterPepCor 11
FitContrasts 12
FitLmerBySample 14
RobustReg 15
SelfContPepSetTestWorkflow 16
TTestwCor 18

Index **20**

`AggLimmaWorkflow` *Aggregation-based LIMMA workflow*

Description

Given peptide abundance and assignment of peptide sequences to proteins, execute the aggregation-based LIMMA workflow to compute the log2 fold change, p-value, and adjusted p-value of all proteins identified.

Usage

```
AggLimmaWorkflow(
  dat,
  contrasts.par,
  group,
  pep_mapping_tbl,
  covar = NULL,
  method = c("sum", "robreg"),
  logged = c(TRUE, FALSE),
  npep.trend = FALSE,
  eb = TRUE
)
```

Arguments

`dat`	a dataframe or matrix of peptide abundance, or a SummarizedExperiment object where grouping and peptide-protein mapping are provided in colData and rowData, respectively.
`contrasts.par`	group levels to be compared separated by dash (e.g., "B-A" if group B is to be compared against group A)

group	a vector of group levels corresponding to each sample. Alternatively, it can be the column name of the group in colData if dat is a SummarizedExperiment object.
pep_mapping_tbl	a table mapping peptides to proteins. Alternatively, it can be the column name of the protein in rowData if dat is a SummarizedExperiment object.
covar	covariate matrix. Alternatively, it can be the column names of the covariates in colData if dat is a SummarizedExperiment object.
method	method of aggregation. Options including "sum" (summed peptide intensity) and "robreg" (robust regression with M-Estimation).
logged	Boolean variable indicating whether abundance data have been log-transformed
npep.trend	logical, should a number-of-peptide-trend be allowed for the prior variance? Default is constant prior variance.
eb	logical, whether to output the result from the empirical Bayes or ordinary approach.

Value

AggLimmaWorkflow returns a dataframe containing the following columns

feature	unique protein identifier
logFC	log2 fold change
t	t-statistic
P.Value	raw p-value
adj.P.Val	p-value adjusted via the Benjamini-Hochberg method
B	B-statistic (empirical Bayes only)

Author(s)

Junmin Wang

References

Ritchie, ME, Phipson, B, Wu, D, Hu, Y, Law, CW, Shi, W, and Smyth, GK (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Research* 43, e47.

Examples

```
# Generate random peptide data
dat <- 2^matrix(rnorm(3000), ncol = 6)
colnames(dat) <- paste0("Sample", 1:6)
rownames(dat) <- paste0("Peptide", 1:500)

# Generate peptide mapping table
pep_mapping_tbl <- data.frame(peptide = paste0("Peptide", 1:500),
protein = paste0("Protein", rep(1:100, each = 5)))
```

```
# Generate groups and contrasts
group <- c(rep("A", 3), rep("B", 3))
contrasts.par <- "B-A"

AggLimmaWorkflow(dat, contrasts.par = contrasts.par,
group = group,
pep_mapping_tbl = pep_mapping_tbl,
method = "sum",
logged = FALSE)

# Store data as a SummarizedExperiment object; add covariates
library(tibble)
library(SummarizedExperiment)
colData <- data.frame(sample = LETTERS[seq_along(group)], group = group,
sex = c("M", "F", "M", "F", "F", "M"), age = 1:6) |>
column_to_rownames(var = "sample")
rowData <- pep_mapping_tbl |> column_to_rownames(var = "peptide")
dat.nn <- dat
rownames(dat.nn) <- NULL
colnames(dat.nn) <- NULL
dat.se <- SummarizedExperiment(assays = list(int = dat.nn), colData = colData, rowData = rowData)

AggLimmaWorkflow(dat.se, contrasts.par = contrasts.par,
group = "group",
covar = c("sex", "age"),
pep_mapping_tbl = "protein",
method = "sum",
logged = FALSE)
```

AggPeps *Aggregate peptide abundance values*

Description

Given peptide abundance and assignment of peptide sequences to proteins, aggregate peptide abundance values into protein abundance values.

Usage

```
AggPeps(
  dat,
  pep_mapping_tbl,
  method = c("sum", "robreg"),
  logged = c(TRUE, FALSE)
)
```

Arguments

dat	a dataframe or matrix of peptide abundance, or a SummarizedExperiment object where grouping and peptide-protein mapping are provided in colData and rowData, respectively.
pep_mapping_tbl	a table mapping peptides to proteins. Alternatively, it can be the column name of the protein in rowData if dat is a SummarizedExperiment object.
method	method of aggregation. Options including "sum" (summed peptide intensity) and "robreg" (robust regression with M-Estimation).
logged	Boolean variable indicating whether abundance data have been log-transformed

Value

AggPeps returns a list containing a matrix of protein abundance values and a vector of number of peptides

Author(s)

Junmin Wang

Examples

```
# Generate random peptide data
dat <- 2^matrix(rnorm(3000), ncol = 6)
colnames(dat) <- paste0("Sample", 1:6)
rownames(dat) <- paste0("Peptide", 1:500)

# Generate peptide mapping table
pep_mapping_tbl <- data.frame(peptide = paste0("Peptide", 1:500),
protein = paste0("Protein", rep(1:100, each = 5)))

AggPeps(dat, pep_mapping_tbl, method = "sum",
logged = FALSE)

# Store data as a SummarizedExperiment object
library(tibble)
library(SummarizedExperiment)
rowData <- pep_mapping_tbl |> column_to_rownames(var = "peptide")
dat.nn <- dat
rownames(dat.nn) <- NULL
colnames(dat.nn) <- NULL
dat.se <- SummarizedExperiment(assays = list(int = dat.nn), rowData = rowData)

AggPeps(dat.se, pep_mapping_tbl = "protein", method = "sum",
logged = FALSE)
```

`CompPepSetTest` *Competitive peptide set test*

Description

Given peptide-wise t-statistics and assignment of peptides to proteins, conduct peptide set tests to assess differential protein expression.

Usage

```
CompPepSetTest(
  result,
  pep_mapping_tbl,
  stat = c("t", "logFC"),
  cor_coef = 0,
  pepC.estim = c("sd", "mad")
)
```

Arguments

`result`	output from `EnframeContrastsRes`
`pep_mapping_tbl`	a table mapping peptides to proteins. Alternatively, it can be the column name of the protein in rowData if dat is a SummarizedExperiment object.
`stat`	statistics to be used in the peptide set test. Options include "t" (t-statistic) and "logFC" (log2 fold change).
`cor_coef`	inter-peptide correlation coefficient(s)
`pepC.estim`	estimator of the variance of peptide-wise t-statistics not belonging to the protein of interest, i.e., test set. Options include "sd" and "mad". "sd" represents sample standard deviation. "mad" represents sample median absolute deviation.

Value

`CompPepSetTest` returns a dataframe containing the following columns

`protein`	unique protein identifier
`NPeps`	number of peptides
`Correlation`	inter-peptide correlation coefficient
`Direction`	direction of change
`PValue`	raw p-value
`adj.P.Val`	p-value adjusted via the Benjamini-Hochberg method
`logFC`	average log2 fold change of peptides
`Up`	number of upregulated peptides
`Down`	number of downregulated peptides

Author(s)

Junmin Wang

Examples

```
# Generate random peptide data
dat <- 2^matrix(rnorm(3000), ncol = 6)
colnames(dat) <- paste0("Sample", 1:6)
rownames(dat) <- paste0("Peptide", 1:500)

# Generate peptide mapping table
pep_mapping_tbl <- data.frame(peptide = paste0("Peptide", 1:500),
protein = paste0("Protein", rep(1:100, each = 5)))

# Generate groups and contrasts
group <- c(rep("A", 3), rep("B", 3))
contrasts.par <- "B-A"

fit.cont <- FitContrasts(dat, contrasts.par, group)
cont.res <- EnframeContrastsRes(fit.cont)

# Run peptide set test based on t-statistics and standard deviation
CompPepSetTest(cont.res, pep_mapping_tbl, stat = "t",
cor_coef = 0, pepC.estim = "sd")

# Run peptide set test based on log2 fold change and median absolute deviation
CompPepSetTest(cont.res, pep_mapping_tbl, stat = "logFC",
cor_coef = 0, pepC.estim = "mad")
```

CompPepSetTestWorkflow

Competitive Peptide Set Test Workflow

Description

Given peptide abundance and assignment of peptide sequences to proteins, execute the competitive peptide set test workflow to compute the log2 fold change, p-value, and adjusted p-value of all proteins identified.

Usage

```
CompPepSetTestWorkflow(
  dat,
  contrasts.par,
  group,
  pep_mapping_tbl,
  covar = NULL,
  stat = c("t", "logFC"),
```

```
  correlated = FALSE,
  equal.correlation = FALSE,
  pepC.estim = c("sd", "mad"),
  logged = FALSE
)
```

Arguments

dat	a dataframe or matrix of peptide abundance (row names should be peptide sequences or peptide IDs), or a SummarizedExperiment object where grouping and peptide-protein mapping are provided in colData and rowData, respectively.
contrasts.par	group levels to be compared separated by dash (e.g., "B-A" if group B is to be compared against group A)
group	a vector of group levels corresponding to each sample. Alternatively, it can be the column name of the group in colData if dat is a SummarizedExperiment object.
pep_mapping_tbl	a table mapping peptides to proteins (it should include two columns named "peptide" and "protein"). Alternatively, it can be the column name of the protein in rowData if dat is a SummarizedExperiment object.
covar	covariate matrix. Alternatively, it can be the column names of the covariates in colData if dat is a SummarizedExperiment object.
stat	statistics to be used in the peptide set test. Options include "t" (t-statistic) and "logFC" (log2 fold change).
correlated	Boolean variable indicating whether peptides are assumed to be correlated. If correlated, inter-peptide correlation will be estimated.
equal.correlation	Boolean variable indicating whether all pairwise inter-peptide correlation coefficients are assumed to be equal within a protein. If true, the mixed model approach will be applied; otherwise, the approach described in Wu and Smyth (2012), *Nucleic Acids Research* will be applied. Note that this parameter matters only if "correlated" is set to true.
pepC.estim	estimator of the variance of peptide-wise t-statistics not belonging to the protein of interest, i.e., test set. Options include "sd" and "mad". "sd" represents sample standard deviation. "mad" represents sample median absolute deviation.
logged	Boolean variable indicating whether abundance data have been log-transformed

Value

`CompPepSetTestWorkflow` returns a dataframe containing the following columns

protein	unique protein identifier
NPeps	number of peptides
Correlation	inter-peptide correlation coefficient
Direction	direction of change
PValue	raw p-value

adj.P.Val	p-value adjusted via the Benjamini-Hochberg method
logFC	average log2 fold change of peptides
Up	number of upregulated peptides
Down	number of downregulated peptides

Author(s)

Junmin Wang

References

Wu, D, and Smyth, GK (2012). Camera: a competitive gene set test accounting for inter-gene correlation. *Nucleic Acids Research* 40, e133

Examples

```
# Generate random peptide data
dat <- 2^matrix(rnorm(3000), ncol = 6)
colnames(dat) <- paste0("Sample", 1:6)
rownames(dat) <- paste0("Peptide", 1:500)

# Generate peptide mapping table
pep_mapping_tbl <- data.frame(peptide = paste0("Peptide", 1:500),
protein = paste0("Protein", rep(1:100, each = 5)))

# Generate groups and contrasts
group <- c(rep("A", 3), rep("B", 3))
contrasts.par <- "B-A"

CompPepSetTestWorkflow(dat, contrasts.par = contrasts.par,
group = group,
pep_mapping_tbl = pep_mapping_tbl,
stat = "t",
correlated = TRUE,
equal.correlation = TRUE,
pepC.estim = "mad",
logged = FALSE)

# Store data as a SummarizedExperiment object; add covariates
library(tibble)
library(SummarizedExperiment)
colData <- data.frame(sample = LETTERS[seq_along(group)], group = group,
sex = c("M", "F", "M", "F", "F", "M"), age = 1:6) |>
column_to_rownames(var = "sample")
rowData <- pep_mapping_tbl |> column_to_rownames(var = "peptide")
dat.nn <- dat
rownames(dat.nn) <- NULL
colnames(dat.nn) <- NULL
dat.se <- SummarizedExperiment(assays = list(int = dat.nn), colData = colData, rowData = rowData)

CompPepSetTestWorkflow(dat.se, contrasts.par = contrasts.par,
```

```
group = "group",
pep_mapping_tbl = "protein",
covar = c("sex", "age"),
stat = "t",
correlated = TRUE,
equal.correlation = TRUE,
pepC.estim = "mad",
logged = FALSE)
```

`EnframeContrastsRes` *Enframe result of LIMMA analysis*

Description

Convert result of LIMMA analysis into a dataframe.

Usage

```
EnframeContrastsRes(eBayes.fit, eb = TRUE)
```

Arguments

`eBayes.fit`	output from `FitContrasts`. See ?`limma::eBayes` for details.
`eb`	logical, whether to output the result from the empirical Bayes or ordinary approach.

Value

`EnframeContrastsRes` returns a dataframe containing the following columns

`feature`	unique feature identifier
`logFC`	log2 fold change
`t`	t-statistic
`P.Value`	raw p-value
`adj.P.Val`	p-value adjusted via the Benjamini-Hochberg method
`B`	B-statistic (empirical Bayes only)

Author(s)

Junmin Wang

Examples

```
# Generate random peptide data
dat <- 2^matrix(rnorm(3000), ncol = 6)
colnames(dat) <- paste0("Sample", 1:6)
rownames(dat) <- paste0("Peptide", 1:500)

# Generate groups and contrasts
group <- c(rep("A", 3), rep("B", 3))
contrasts.par <- "B-A"

fit.cont <- FitContrasts(dat, contrasts.par, group)
EnframeContrastsRes(fit.cont)
```

`EstimInterPepCor` *Estimation of inter-peptide correlation*

Description

Given peptide abundance and assignment of peptide sequences to proteins, estimate inter-peptide correlation coefficient for each protein via the mixed model approach or approach described in Wu and Smyth (2012), *Nucleic Acids Research*.

Usage

```
EstimInterPepCor(
  dat,
  design,
  pep_mapping_tbl,
  equal.correlation = FALSE,
  logged = FALSE
)
```

Arguments

`dat`	a dataframe or matrix of peptide abundance, or a SummarizedExperiment object where grouping and peptide-protein mapping are provided in colData and rowData, respectively.
`design`	design matrix
`pep_mapping_tbl`	a table mapping peptides to proteins. Alternatively, it can be the column name of the protein in rowData if dat is a SummarizedExperiment object.
`equal.correlation`	Boolean variable indicating whether all pairwise inter-peptide correlation coefficients are assumed to be equal within a protein. If true, the mixed model approach will be applied; otherwise, the approach described in Wu and Smyth (2012), *Nucleic Acids Research* will be applied. In either case, only non-negative mean correlations are allowed.

`logged` Boolean variable indicating whether abundance data have been log-transformed

Value

`EstimInterPepCor` returns a numeric vector of inter-peptide correlation coefficients (one value for each protein).

Author(s)

Junmin Wang and Steven Novick

References

Wu, D, and Smyth, GK (2012). Camera: a competitive gene set test accounting for inter-gene correlation. *Nucleic Acids Research* 40, e133

Examples

```
# Generate random peptide data
dat <- 2^matrix(rnorm(540), ncol = 6)
colnames(dat) <- paste0("Sample", 1:6)
rownames(dat) <- paste0("Peptide", 1:90)

# Generate peptide mapping table
pep_mapping_tbl <- data.frame(peptide = paste0("Peptide", 1:90),
protein = paste0("Protein", rep(1:30, each = 3)))

# Generate design matrix
group <- c(rep("A", 3), rep("B", 3))
group <- factor(group)
design <- stats::model.matrix(~ 0 + group)

EstimInterPepCor(dat, design, pep_mapping_tbl,
equal.correlation = TRUE, logged = FALSE)
```

`FitContrasts` *Empirical Bayes moderated t-test*

Description

Fit a linear model to feature abundance and compute moderated t-statistics via the empirical Bayes method.

Usage

```
FitContrasts(
  dat,
  contrasts.par,
  group,
  covar = NULL,
  logged = FALSE,
  NPeptide = NULL
)
```

Arguments

dat	a dataframe or matrix of feature (e.g., peptide, protein) abundance
contrasts.par	group levels to be compared separated by dash (e.g., "B-A" if group B is to be compared against group A)
group	list of group levels corresponding to each sample. The order of group levels needs to match that of samples in the feature abundance table.
covar	covariate matrix
logged	Boolean variable indicating whether data have been log-transformed
NPeptide	numeric vector indicating number of peptides aggregated for each protein. logNPeptide will be passed to limma-trend. Constant prior variance if null.

Value

`FitContrasts` returns an object of class `MArrayLM`. See `?limma::eBayes` for details.

Author(s)

Junmin Wang

References

Ritchie, ME, Phipson, B, Wu, D, Hu, Y, Law, CW, Shi, W, and Smyth, GK (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Research* 43, e47.

Examples

```
# Generate random peptide data
dat <- 2^matrix(rnorm(3000), ncol = 6)
colnames(dat) <- paste0("Sample", 1:6)
rownames(dat) <- paste0("Peptide", 1:500)

# Generate groups and contrasts
group <- c(rep("A", 3), rep("B", 3))
contrasts.par <- "B-A"

# Run moderated t-test without covariates
```

```
FitContrasts(dat, contrasts.par, group)

# Run moderated t-test with covariates
covar <- matrix(c(1:6, 0, 1, 0, 1, 1, 0), nrow = 6, ncol = 2, byrow = FALSE)
FitContrasts(dat, contrasts.par, group, covar = covar)
```

`FitLmerBySample` *Fit a linear mixed model*

Description

Fit a linear mixed model to the abundance of peptides belonging to one protein and compute the correlation coefficient based on variance components. Sample is treated as a random effect in the mixed model.

Usage

```
FitLmerBySample(y, design)
```

Arguments

`y`	a matrix of log2-transformed peptide abundance for one protein
`design`	design matrix

Value

`FitLmerBySample` returns the estimated inter-peptide correlation coefficient.

Author(s)

Junmin Wang and Steven Novick

Examples

```
y <- matrix(rnorm(1000*6), 1000, 6)
design <- cbind(Intercept = 1, Group = c(0, 0, 0, 1, 1, 1))

FitLmerBySample(y, design)
```

RobustReg *Robust Regression*

Description

Estimate protein abundance by fitting a linear model to peptide abundance via robust regression.

Usage

```
RobustReg(dat, logged = FALSE)
```

Arguments

dat	a dataframe or matrix of peptide abundance, or a SummarizedExperiment object where grouping and peptide-protein mapping are provided in colData and rowData, respectively.
logged	Boolean variable indicating whether abundance data have been log-transformed

Value

RobustReg returns a numeric vector of estimated protein abundance.

Author(s)

Junmin Wang

References

Sticker, A, Goeminne, L, Martens, L, and Clement, L (2020). Robust Summarization and Inference in Proteome-wide Label-free Quantification. *Molecular & Cellular Proteomics* 19, 1209-19.

Gatto, L, Rainer, J, and Gibb, S (2021). MsCoreUtils: Core Utils for Mass Spectrometry Data. R packageversion 1.4.0. https://github.com/RforMassSpectrometry/MsCoreUtils

Examples

```
# Generate random peptide data
dat <- 2^matrix(rnorm(600), ncol = 6)
colnames(dat) <- paste0("Sample", 1:6)
rownames(dat) <- paste0("Peptide", 1:100)

RobustReg(dat, logged = FALSE)
```

`SelfContPepSetTestWorkflow` *Self-contained Peptide Set Test Workflow*

Description

Given peptide abundance and assignment of peptide sequences to proteins, execute the self-contained peptide set test workflow to compute the log2 fold change, p-value, and adjusted p-value of all proteins identified.

Usage

```
SelfContPepSetTestWorkflow(
  dat,
  contrasts.par,
  group,
  pep_mapping_tbl,
  covar = NULL,
  logged = FALSE
)
```

Arguments

`dat`	a dataframe or matrix of peptide abundance (row names should be peptide sequences or peptide IDs), or a SummarizedExperiment object where grouping and peptide-protein mapping are provided in colData and rowData, respectively.
`contrasts.par`	group levels to be compared separated by dash (e.g., "B-A" if group B is to be compared against group A)
`group`	a vector of group levels corresponding to each sample. Alternatively, it can be the column name of the group in colData if dat is a SummarizedExperiment object.
`pep_mapping_tbl`	a table mapping peptides to proteins (it should include two columns named "peptide" and "protein"). Alternatively, it can be the column name of the protein in rowData if dat is a SummarizedExperiment object.
`covar`	covariate matrix. Alternatively, it can be the column names of the covariates in colData if dat is a SummarizedExperiment object.
`logged`	Boolean variable indicating whether abundance data have been log-transformed

Value

`SelfContPepSetTestWorkflow` returns a dataframe containing the following columns

`protein`	unique protein identifier
`NPeps`	number of peptides

Direction	direction of change
PValue	raw p-value
adj.P.Val	adjusted p-value
logFC	average log2 fold change of peptides
Up	number of upregulated peptides
Down	number of downregulated peptides

Author(s)

Junmin Wang

References

Wu, D, Lim, E, Francois Vaillant, F, Asselin-Labat, M-L, Visvader, JE, and Smyth, GK (2010). ROAST: rotation gene set tests for complex microarray experiments. *Bioinformatics* 26, 2176-2182

Examples

```
# Generate random peptide data
dat <- 2^matrix(rnorm(3000), ncol = 6)
colnames(dat) <- paste0("Sample", 1:6)
rownames(dat) <- paste0("Peptide", 1:500)

# Generate peptide mapping table
pep_mapping_tbl <- data.frame(peptide = paste0("Peptide", 1:500),
protein = paste0("Protein", rep(1:100, each = 5)))

# Generate groups and contrasts
group <- c(rep("A", 3), rep("B", 3))
contrasts.par <- "B-A"

SelfContPepSetTestWorkflow(dat, contrasts.par = contrasts.par,
group = group,
pep_mapping_tbl = pep_mapping_tbl,
logged = FALSE)

# Store data as a SummarizedExperiment object; add covariates
library(tibble)
library(SummarizedExperiment)
colData <- data.frame(sample = LETTERS[seq_along(group)], group = group,
sex = c("M", "F", "M", "F", "F", "M"), age = 1:6) |>
column_to_rownames(var = "sample")
rowData <- pep_mapping_tbl |> column_to_rownames(var = "peptide")
dat.nn <- dat
rownames(dat.nn) <- NULL
colnames(dat.nn) <- NULL
dat.se <- SummarizedExperiment(assays = list(int = dat.nn), colData = colData, rowData = rowData)

SelfContPepSetTestWorkflow(dat.se, contrasts.par = contrasts.par,
group = "group",
```

```
pep_mapping_tbl = "protein",
covar = c("sex", "age"),
logged = FALSE)
```

`TTestwCor` *Two-sample t-test accounting for inter-peptide correlation*

Description

Test whether peptides belonging to the same protein are differentially expressed relative to the rest of the peptidome, accounting for inter-peptide correlation. This function is adapted from `cameraPR()` in LIMMA R package (Wu and Smyth (2012), *Nucleic Acids Research*).

Usage

```
TTestwCor(statistic, index, inter.pep.cor, pepC.estim = c("sd", "mad"))
```

Arguments

`statistic`	a numeric vector of peptide-wise t-statistics.
`index`	an index vector or a list of index vectors. `statistic[index]` selects corresponding rows in the protein of interest, i.e., test set(s).
`inter.pep.cor`	a numeric vector of inter-peptide correlation coefficients within the protein of interest, i.e., test set(s).
`pepC.estim`	estimator of the variance of peptide-wise t-statistics not belonging to the protein of interest, i.e., test set. Options include "sd" and "mad". "sd" represents sample standard deviation. "mad" represents sample median absolute deviation.

Value

`TTestwCor` returns a dataframe in which each row represents a protein of interest, i.e., test set. Columns include

`NPeps`	number of peptides
`Correlation`	inter-peptide correlation coefficient
`Direction`	direction of change
`PValue`	raw p-value
`adj.P.Val`	p-value adjusted via the Benjamini-Hochberg method

Author(s)

Junmin Wang

References

Wu, D, and Smyth, GK (2012). Camera: a competitive gene set test accounting for inter-gene correlation. *Nucleic Acids Research* 40, e133

Examples

```
y <- matrix(rnorm(1000 * 6), 1000, 6)
design <- cbind(Intercept = 1, Group = c(0, 0, 0, 1, 1, 1))

# First set of 20 genes are genuinely differentially expressed
index1 <- 1:20
y[index1, 4:6] <- y[index1, 4:6]+1

fit <- limma::eBayes(limma::lmFit(y, design))
TTestwCor(fit$t[, 2], index = index1,
inter.pep.cor = 0,
pepC.estim = "sd")
```

Index

AggLimmaWorkflow, 2
AggPeps, 4

CompPepSetTest, 6
CompPepSetTestWorkflow, 7

EnframeContrastsRes, 10
EstimInterPepCor, 11

FitContrasts, 12
FitLmerBySample, 14

RobustReg, 15

SelfContPepSetTestWorkflow, 16

TTestwCor, 18