Package 'HERON'

November 12, 2026

Type Package

Date 2024-08-22

Title Hierarchical Epitope pROtein biNding

Version 1.10.0

Description HERON is a software package for analyzing peptide binding array data. In addition to identifying significant binding probes, HERON also provides functions for finding epitopes (string of consecutive peptides within a protein). HERON also calculates significance on the probe, epitope, and protein level by employing meta p-value methods. HERON is designed for obtaining calls on the sample level and calculates fractions of hits for different conditions.

License GPL (>= 3)

URL https://github.com/Ong-Research/HERON

BugReports https://github.com/Ong-Research/HERON/issues

Encoding UTF-8

LazyData false

Imports matrixStats, stats, data.table, harmonicmeanp, metap, cluster, spdep, Matrix, limma, methods

RoxygenNote 7.3.1

biocViews Microarray, Software

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Depends R (>= 4.4.0), SummarizedExperiment (>= 1.1.6), GenomicRanges, IRanges, S4Vectors

git_url https://git.bioconductor.org/packages/HERON

git_branch RELEASE_3_23

git_last_commit 4820dd1

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Sean McIlwain [aut, cre] (ORCID:
<https://orcid.org/0000-0002-3820-8400>),
Irene Ong [aut] (ORCID: <https://orcid.org/0000-0002-9353-6941>)

Maintainer Sean McIlwain <sean.mcilwain@wisc.edu>

Contents

HERON-package . . . 3
addSequenceAnnotations . . . 3
calcCombPValues . . . 4
calcEpitopePValues . . . 5
calcProbePValuesTPaired . . . 6
calcProbePValuesTUnpaired . . . 7
calcProbePValuesWUnpaired . . . 8
calcProteinPValues . . . 8
catSequences . . . 9
convertSequenceDSToProbeDS . . . 10
findBlocksProbeT . . . 11
findBlocksT . . . 11
findEpitopeSegments . . . 12
getEpitopeID . . . 13
getEpitopeIDsToProbeIDs . . . 13
getEpitopeProbeIDs . . . 14
getEpitopeProtein . . . 15
getEpitopeStart . . . 15
getEpitopeStop . . . 16
getKofN . . . 16
getProteinLabel . . . 17
getProteinStart . . . 17
getProteinTiling . . . 18
heffron2021_wuhan . . . 18
HERONEpitopeDataSet-class . . . 19
HERONProbeDataSet-class . . . 20
HERONProteinDataSet-class . . . 20
HERONSequenceDataSet-class . . . 21
log2Transform . . . 21
makeEpitopeCalls . . . 22
makeProbeCalls . . . 23
makeProteinCalls . . . 23
min_max . . . 24
oneHitEpitopes . . . 25
oneHitProbes . . . 25
oneProbeEpitopes . . . 26

probeHitSupported 26
pvalue_to_zscore 27
quantileNormalize 27
smoothProbeDS 28

Index **29**

`HERON-package` *HERON: Hierarchical Epitope pROtein biNding*

Description

HERON is a software package for analyzing peptide binding array data. In addition to identifying significant binding probes, HERON also provides functions for finding epitopes (string of consecutive peptides within a protein). HERON also calculates significance on the probe, epitope, and protein level by employing meta p-value methods. HERON is designed for obtaining calls on the sample level and calculates fractions of hits for different conditions.

Author(s)

Maintainer: Sean McIlwain <sean.mcilwain@wisc.edu> (ORCID)

Authors:

- Irene Ong <irene.ong@wisc.edu> (ORCID)

See Also

Useful links:

- https://github.com/Ong-Research/HERON
- Report bugs at https://github.com/Ong-Research/HERON/issues

---

`addSequenceAnnotations` *Add Sequence Annotations for Epitopes*

Description

Add Sequence Annotations for Epitopes

Usage

```
addSequenceAnnotations(eds)
```

Arguments

`eds`	HERONEpitopeDataSet with probe_meta in metadata()

Value

HERONEpitopeDataSet with the rowData() set with sequence annotations

Examples

```
data(heffron2021_wuhan)
pval_seq_res <- calcCombPValues(heffron2021_wuhan)
pval_pr_res <- convertSequenceDSToProbeDS(pval_seq_res)
calls_res <- makeProbeCalls(pval_pr_res)
segments_res <- findEpitopeSegments(calls_res, "unique")
epval_res <- calcEpitopePValues(calls_res, segments_res)
epval_res <- addSequenceAnnotations(epval_res)
```

---

calcCombPValues *Calculate p-values using the "exprs" assay*

Description

Calculate p-values using the "exprs" assay

Usage

```
calcCombPValues(
  obj,
  colData_in = NULL,
  d_sd_shift = NA,
  d_abs_shift = NA,
  d_paired = FALSE,
  g_sd_shift = 0,
  use = "tz",
  p_adjust_method = "BH"
)
```

Arguments

obj	HERONSequenceDataSet or HERONProbeDataSet
colData_in	optional column DataFrame (default: NULL => colData(obj)))
d_sd_shift	standard deviation shift for differential test
d_abs_shift	absolute shift for differential test
d_paired	run paired analysis
g_sd_shift	standard deviation shift for global test
use	use global-test ("z"), differential-test using t.test ("t"), differential-test using wilcox ("w"), or both global and differential ("tz")
p_adjust_method	method for adjusting p-values

Value

HERONSequenceDataSet/HERONProbeDataSet with the pvalue assay added

Examples

```
data(heffron2021_wuhan)
seq_pval_res <- calcCombPValues(heffron2021_wuhan)
```

`calcEpitopePValues` *Calculate epitope-level p-values*

Description

Calculate epitope-level p-values

Usage

```
calcEpitopePValues(
  probe_pds,
  epitope_ids,
  metap_method = "wmax1",
  p_adjust_method = "BH"
)
```

Arguments

`probe_pds`	HERONProbeDataSet with the "pvalue" assay
`epitope_ids`	vector of epitope ids
`metap_method`	meta p-value method to use (see below)
`p_adjust_method`	what p.adjust method to use.

Details

The meta p-value methods supported by `calcEpitopePValues` are: min_bonf*, min*, max*, fischer/sumlog, hmp/harmonicmeanp, wilkinsons_min1/tippets, wilkinsons_min2/wmin2, wilkinsons_min3, wilkinsons_min4, wilkinsons_min5, wilkinsons_max1/wmax1, wilkinsons_max2/wmax2, and cct.

When choosing a p-value method, keep in mind that the epitope p-value should be one that requires most of the probe p-values to be small (e.g. *wmax1*) Other p-value methods such as the*cct* and the *hmp* have been shown to be more accurate with p-value that have dependencies.

Value

HERONEpitopeDataSet with "pvalue" and "padj" assays

See Also

[stats::p.adjust()] for p_adjust_parameter.

Examples

```
data(heffron2021_wuhan)
pval_seq_res <- calcCombPValues(heffron2021_wuhan)
pval_pr_res <- convertSequenceDSToProbeDS(pval_seq_res)
calls_res <- makeProbeCalls(pval_pr_res)
segments_res <- findEpitopeSegments(calls_res, "unique")
epval_res <- calcEpitopePValues(calls_res, segments_res)
```

---

calcProbePValuesTPaired

Calculate Probe p-values using a differential paired t-test

Description

Calculate Probe p-values using a differential paired t-test

Usage

```
calcProbePValuesTPaired(
  probe_mat,
  colData_in,
  sd_shift = NA,
  abs_shift = NA,
  debug = FALSE
)
```

Arguments

probe_mat	numeric matrix or data.frame of values
colData_in	design data.frame
sd_shift	standard deviation shift to use when calculating p-values. Either sd_shift or abs_shift should be set
abs_shift	absolute shift to use when calculating p-values.
debug	print debugging information

Value

matrix of p-values on the post columns defined in the colData matrix. Attributes of the matrix are:

pars - data.frame parameters used in the paired t-test for each row (e.g. df, sd)

mapping - data.frame of mapping used for pre-post column calculation diff_mat - data.frame containing the post-pre differences for each sample (column) and probe (row)

Examples

```
data(heffron2021_wuhan)
colData_wu <- colData(heffron2021_wuhan)
pre_idx = which(colData_wu$visit == "pre")
## Make some samples paired
colData_post = colData_wu[colData_wu$visit == "post",]
new_ids = rownames(colData_post)[seq_len(5)]
colData_wu$ptid[pre_idx[seq_len(5)]] = new_ids
exprs <- assay(heffron2021_wuhan, "exprs")
pval_res <- calcProbePValuesTPaired(exprs, colData_wu)
```

---

calcProbePValuesTUnpaired

Calculate Probe p-values using a differential unpaired t-test

---

Description

Calculate Probe p-values using a differential unpaired t-test

Usage

```
calcProbePValuesTUnpaired(probe_mat, colData_in, sd_shift = NA, abs_shift = NA)
```

Arguments

probe_mat	numeric matrix or data.frame of values
colData_in	design data.frame
sd_shift	standard deviation shift to use when calculating p-values Either sd_shift or abs_shift should be set
abs_shift	absolute shift to use when calculating p-values

Value

matrix of p-values on the post columns defined in the colData matrix

Examples

```
data(heffron2021_wuhan)
colData_wu <- colData(heffron2021_wuhan)
pval_res <- calcProbePValuesTUnpaired(assay(heffron2021_wuhan), colData_wu)
```

calcProbePValuesWUnpaired *Calculate Probe p-values using a two-sample wilcoxon test*

Description

Calculate Probe p-values using a two-sample wilcoxon test

Usage

```
calcProbePValuesWUnpaired(probe_mat, colData_in, exact = NULL, abs_shift = 0)
```

Arguments

probe_mat	numeric matrix or data.frame of values
colData_in	design data.frame
exact	a logical indicating whether an exact p-value should be computed (see wilcox.test for details)
abs_shift	absolute shift to use when calculating p-values

Value

matrix of p-values on the post columns defined in the colData matrix

Examples

```
data(heffron2021_wuhan)
colData_wu <- colData(heffron2021_wuhan)
pval_res <- calcProbePValuesWUnpaired(assay(heffron2021_wuhan), colData_wu)
```

calcProteinPValues *Calculate protein-level p-values*

Description

Calculate protein-level p-values

Usage

```
calcProteinPValues(epitope_ds, metap_method = "wmin1", p_adjust_method = "BH")
```

Arguments

epitope_ds	HERONEpitopeDataSet with the "pvalue" assay
metap_method	meta p-value method to use
p_adjust_method	p.adjust method to use

Details

see calcEpitopePValues for a list of meta p-value methods supported by HERON. the protein should be one that requires at least one of the epitope p-values to be small (e.g. wmax1).

Value

HERONProteinDataSet with the "pvalue" and "padj" assays

See Also

[stats::p.adjust()] for p_adjust_parameter.

[calcEpitopePValues()] for meta p-value methods

Examples

```
data(heffron2021_wuhan)
pval_seq_res <- calcCombPValues(heffron2021_wuhan)
pval_pr_res <- convertSequenceDSToProbeDS(pval_seq_res)
calls_res <- makeProbeCalls(pval_pr_res)
segments_res <- findEpitopeSegments(calls_res, "unique")
epval_res <- calcEpitopePValues(calls_res, segments_res)
ppval_res <- calcProteinPValues(epval_res)
```

---

catSequences *Concatenate sequences together based upon their start positions. Assumes the probe sequences have an overlap.*

Description

Concatenate sequences together based upon their start positions. Assumes the probe sequences have an overlap.

Usage

```
catSequences(positions, sequences)
```

Arguments

positions	start positions of probes in protein
sequences	probe sequences of probes

Value

concatenated sequence (character)

Examples

```
positions <- c(1,2)
sequences <- c("MSGSASFEGGVFSPYL", "SGSASFEGGVFSPYLT")
catSequences(positions, sequences)
```

---

convertSequenceDSToProbeDS *Convert HERONSequenceDataSet to HERONProbeDataSet*

Description

Convert HERONSequenceDataSet to HERONProbeDataSet

Usage

```
convertSequenceDSToProbeDS(seq_ds, probe_meta)
```

Arguments

seq_ds	a HERONSequenceDataSet object
probe_meta	optional data.frame with the PROBE_SEQUENCE, PROBE_ID columns the probe meta data frame can be provided within the metadata()$probe_meta or as a argument to the function. The argument supersedes the metadata list.

Value

HERONProbeDataSet

Examples

```
data(heffron2021_wuhan)
probe_ds <- convertSequenceDSToProbeDS(heffron2021_wuhan)
probe_meta <- metadata(heffron2021_wuhan)$probe_meta
probe_ds <- convertSequenceDSToProbeDS(heffron2021_wuhan, probe_meta)
```

findBlocksProbeT *Find Blocks of consecutive probes*

Description

This function will find blocks of consecutive probes within the passed probe parameter

Usage

```
findBlocksProbeT(
  probes,
  protein_tiling,
  proteins = getProteinLabel(probes),
  starts = getProteinStart(probes)
)
```

Arguments

probes	vector of probe identifiers of the format c(Prot1;1, ... Prot1;10)
protein_tiling	tiling of the associated proteins
proteins	associated proteins to probes (cache speed up)
starts	associated starts from probes (cache speed up)

Value

data.frame with the Protein, Start, Stop, and Number.Of.Probes columns

Examples

```
findBlocksProbeT(c("A;1", "A;2", "A;3", "B;2", "B;3", "C;10", "A;5", "A;6"))
```

findBlocksT *Find consecutive probes*

Description

Find consecutive probes

Usage

```
findBlocksT(prot_df, protein_tiling)
```

Arguments

prot_df	data.frame with the Protein and Starting position of the probe
protein_tiling	tiling for information for each protein

Value

data.frame with the Protein, Start, Stop, and Number.Of.Probes columns

Examples

```
probes = c("A;1","A;2","A;3", "A;5","A;6", "A;8")
prot_df = data.frame(
    Protein = getProteinLabel(probes),
    Pos = getProteinStart(probes)
)
findBlocksT(prot_df)
```

---

findEpitopeSegments *Find Epitopes from probe stats and calls.*

Description

Find Epitopes from probe stats and calls.

Usage

```
findEpitopeSegments(
  PDS_obj,
  segment_method = "unique",
  segment_score_type = "binary",
  segment_dist_method = "hamming",
  segment_cutoff = "silhouette"
)
```

Arguments

PDS_obj	HERONProbeDataSet with pvalues and calls in the assay
segment_method	which epitope finding method to use (binary or zscore, applies for hclust or skater)
segment_score_type	which type of scoring to use for probes
segment_dist_method	what kind of distance score method to use
segment_cutoff	for clustering methods, what cutoff to use (either numeric value or 'silhouette')

Value

a vector of epitope identifiers or segments found

Examples

```
data(heffron2021_wuhan)
seq_pval_res <- calcCombPValues(heffron2021_wuhan)
pr_pval_res <- convertSequenceDSToProbeDS(seq_pval_res)
pr_calls_res <- makeProbeCalls(pr_pval_res)
segments_res <- findEpitopeSegments(pr_calls_res)
```

`getEpitopeID` *Create EpitopeID from protein, first and last probes*

Description

Create EpitopeID from protein, first and last probes

Usage

```
getEpitopeID(protein, start, stop)
```

Arguments

`protein`	vector of proteins
`start`	vector of first probe protein start positions
`stop`	vector of last probe protein start positions

Value

vector of epitope ids

Examples

```
getEpitopeID("A", 1, 2)
```

`getEpitopeIDsToProbeIDs`
Get probe ids from a vector of epitope ids

Description

Get probe ids from a vector of epitope ids

Usage

```
getEpitopeIDsToProbeIDs(epitope_ids, tiling = 1)
```

Arguments

epitope_ids	vector of epitope identifiers
tiling	tling of probes across proteins

Value

data.frame of epitope_to_probe mappings

Examples

```
getEpitopeIDsToProbeIDs(c("A_1_5","C_8_12"))
```

---

getEpitopeProbeIDs *Get the vector of probes from an epitope id*

Description

Get the vector of probes from an epitope id

Usage

```
getEpitopeProbeIDs(epitope_id, tiling = 1)
```

Arguments

epitope_id	EpitopeID to obtain probes from
tiling	Tiling of the probes across the protein (default 1)

Value

vector of probe_ids that are contained within the epitope

Examples

```
getEpitopeProbeIDs("A_1_5")
```

getEpitopeProtein *Obtain Protein Id from Epitope ID*

Description

Format of EpitopeID is A_B_C, where A is the protein label B is the protein start position of the first probe in the epitope and C is the protein start position of the last probe in the epitope.

Usage

```
getEpitopeProtein(epitope_ids)
```

Arguments

epitope_ids	vector of epitope identifier character strings

Value

vector of protein labels

Examples

```
getEpitopeProtein("Prot1_1_5")
```

getEpitopeStart *Obtain first probe's protein start position from Epitope ID*

Description

Obtain first probe's protein start position from Epitope ID

Usage

```
getEpitopeStart(epitope_ids)
```

Arguments

epitope_ids	vector of epitope ids

Value

vector of integers indicating first probe start positions in the epitope(s)

Examples

```
getEpitopeStart("Prot1_1_5")
```

getEpitopeStop *Obtain last probe's protein start position from EpitopeID*

Description

Obtain last probe's protein start position from EpitopeID

Usage

```
getEpitopeStop(epitope_ids)
```

Arguments

epitope_ids	vector of epitope ids

Value

vector of integers indicating the last probe protein start position

Examples

```
getEpitopeStop("Prot1_1_5")
```

getKofN *Get K of N statistics from an experiment with padj and calls*

Description

Calculates the number of samples (K), the frequency of samples (F), and the percentage of samples (P) called. If the colData DataFrame contains a condition column with at least two conditions, then a K, F, and P is calculated for each condition and the results are reported as separate columns.

Usage

```
getKofN(obj)
```

Arguments

obj	HERON Dataset with a "calls" assay

Value

DataFrame with K (#calls), F (fraction calls), P (

Examples

```
data(heffron2021_wuhan)
seq_pval_res <- calcCombPValues(heffron2021_wuhan)
pr_pval_res <- convertSequenceDSToProbeDS(seq_pval_res)
pr_calls_res <- makeProbeCalls(pr_pval_res)
getKofN(pr_calls_res)
```

---

getProteinLabel *Get Protein Label from Probe*

Description

Get Protein Label from Probe

Usage

```
getProteinLabel(probes)
```

Arguments

probes	vector of probes (i.e. c("A;1", "A;2"))

Value

vector of strings indicating the protein associated with the respective probes

Examples

```
getProteinLabel("A;1")
getProteinLabel("B;2")
getProteinLabel(c("A;1","B;2"))
```

---

getProteinStart *Get the amino-acid starting position of the probe within the protein.*

Description

Get the amino-acid starting position of the probe within the protein.

Usage

```
getProteinStart(probes)
```

Arguments

probes	vector of probes (i.e. c("A;1", "A;2"))

Value

starting locations of the probes with their associated proteins

Examples

```
getProteinStart("A;1")
getProteinStart("B;2")
getProteinStart(c("A;1","B;2"))
```

`getProteinTiling` *Get Protein Tiling*

Description

Given a set of probes, estimate the tiling of the probes across the protein. Usually, you will want to calculate this on all the probes available in the dataset.

Usage

```
getProteinTiling(probes, return.vector = TRUE)
```

Arguments

`probes`	vector of probes (i.e. A;1, A;2)
`return.vector`	Return result as vector or return as data.frame

Value

For each protein, the estimating tiling (spacing) of the probes across the amino acid sequence.

Examples

```
getProteinTiling(c("A;1","A;2","A;3", "B;2","B;3", "C;1", "C;3"))
```

`heffron2021_wuhan` *SARS CoV-2 Wuhan Peptide Binding Array Data*

Description

A subset of data from the paper https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8245122/ publication.

Usage

```
data(heffron2021_wuhan)
```

Format

‘heffron2021_wuhan‘ A HERONSequenceDataSet with and "exprs" assay DataFrame with 1945 rows and 60 columns. Each column is a pre-processed binding signal from a serum sample peptide array set for the SARS-CoV-2. The matrix is a subset of the full matrix and contains sequences from the membrane, envelope, surface (spike), and nucleocapsid proteins.

The metadata()$probe_meta is a data frame with 1945 rows and 6 columns. The columns are POSITION - starting position of probe within protein, PROBE_SEQUENCE - amino acid sequence of probe, SEQ_ID - protein identifier SEQ_NAME - name of protein, PROBE_ID - combination of protein identifier and starting position, e.g. prot1;5.

The colData() is a DataFrame with 60 rows and 2 columns. The columns are SampleName - name of the sample, visit - either pre or post, ptid - subject id, and condition - all COVID

Value

HERONSequenceDataSet

Source

<https://github.com/Ong-Research/UW_Adult_Covid-19>

HERONEpitopeDataSet-class

HERONEpitopeDataSet object and constructors

Description

`HERONEpitopeDataSet` is a subclass of `SummarizedExperiment` used to hold assay information on the epitope-level

Usage

```
HERONEpitopeDataSet(pvalue, ...)
```

Arguments

`pvalue`	calculate epitope p-value matrix
`...`	arguments provided to `SummarizedExperiment`, including metadata

Value

HERONEpitopeDataSet object

Examples

```
pval <- matrix(runif(100),ncol=4)
HERONEpitopeDataSet(pvalue = pval)
```

HERONProbeDataSet-class

HERONProbeDataSet object and constructors

Description

HERONProbeDataSet is a subclass of RangedSummarizedExperiment used to hold assay information on the probe level

Usage

```
HERONProbeDataSet(...)
```

Arguments

...	arguments provided to SummarizedExperiment, including metadata.

Value

HERONProbeDataSet object

Examples

```
pds <- HERONProbeDataSet()
```

HERONProteinDataSet-class

HERONProteinDataSet object and constructors

Description

HERONProteinDataSet is a subclass of SummarizedExperiment used to hold assay information on the protein-level

Usage

```
HERONProteinDataSet(pvalue, ...)
```

Arguments

pvalue	calculated protein p-value matrix
...	arguments provided to SummarizedExperiment, including metadata

Value

HERONProteinDataSet object

Examples

```
pval <- matrix(runif(100), ncol=4)
HERONProteinDataSet(pvalue = pval)
```

HERONSequenceDataSet-class *HERONSequenceDataSet object and constructors*

Description

HERONSequenceDataSet is a subclass of SummarizedExperiment, used to store the expression values, intermediate calculations, and results of a differential binding code on the seeuqnce-level.

Usage

```
HERONSequenceDataSet(exprs, ...)
```

Arguments

exprs	binding values with rows as sequences and columns as samples
...	arguments provided to SummarizedExperiment, including metadata metadata can contain a probe DataFrame, that maps sequences (column PROBE_SEQUENCE) to probe identifiers (column PROBE_ID)

Value

HERONSequenceDataSet object

Examples

```
exprs <- matrix(seq_len(100),ncol=4)
colnames(exprs) <- c("C1", "C2", "C3", "C4")
sds <- HERONSequenceDataSet(exprs = exprs)
```

---

log2Transform *log2 transform the "exprs" assay*

Description

log2 transform the "exprs" assay

Usage

```
log2Transform(se)
```

Arguments

se	SummarizedExperiment with "exprs" assay

Value

SummarizedExperiment with "exprs" assay log2 transformed

Examples

```
data(heffron2021_wuhan)
assay(heffron2021_wuhan, "exprs") <- 2^assay(heffron2021_wuhan, "exprs")
res <- log2Transform(heffron2021_wuhan)
```

---

makeEpitopeCalls *Make Epitope Calls*

Description

Make Epitope Calls

Usage

```
makeEpitopeCalls(epi_ds, padj_cutoff = 0.05, one_hit_filter = TRUE)
```

Arguments

epi_ds	HERONEpitopeDataSet with pvalue assay
padj_cutoff	p-value cutoff to use
one_hit_filter	filter one hit epitopes?

Value

HERONEpitopeDataSet with calls assay added

Examples

```
data(heffron2021_wuhan)
seq_pval_res <- calcCombPValues(heffron2021_wuhan)
pr_pval_res <- convertSequenceDSToProbeDS(seq_pval_res)
pr_calls_res <- makeProbeCalls(pr_pval_res)
epi_segments_uniq_res <- findEpitopeSegments(
    PDS_obj = pr_calls_res,
    segment_method = "unique"
)
epi_padj_uniq <- calcEpitopePValues(
    probe_pds = pr_calls_res,
    epitope_ids = epi_segments_uniq_res,
    metap_method = "wilkinsons_max1"
)
makeEpitopeCalls(epi_padj_uniq)
```

makeProbeCalls *Making Probe-level Calls*

Description

makeProbeCalls returns call information on a HERONProbeDataSet using the "padj" assay

Usage

```
makeProbeCalls(pds, padj_cutoff = 0.05, one_hit_filter = TRUE)
```

Arguments

pds	HERONProbeDataSet with the "padj" assay
padj_cutoff	cutoff to use
one_hit_filter	filter out one-hit probes?

Value

HERONProbeDataSet with the "calls" assay added

Examples

```
data(heffron2021_wuhan)
pval_seq_res <- calcCombPValues(heffron2021_wuhan)
pval_probe_res <- convertSequenceDSToProbeDS(pval_seq_res)
calls_res <- makeProbeCalls(pval_probe_res)
```

makeProteinCalls *Make Protein-level Calls*

Description

Make Protein-level Calls

Usage

```
makeProteinCalls(prot_ds, padj_cutoff = 0.05, one_hit_filter = FALSE)
```

Arguments

prot_ds	HERONProteinDataSet with the "padj" assay
padj_cutoff	cutoff to use
one_hit_filter	use the one-hit filter?

Value

HERONProteinDataSet with the "calls" assay added

Examples

```
data(heffron2021_wuhan)
seq_pval_res <- calcCombPValues(heffron2021_wuhan)
pr_pval_res <- convertSequenceDSToProbeDS(seq_pval_res)
pr_calls_res <- makeProbeCalls(pr_pval_res)
epi_segments_uniq_res <- findEpitopeSegments(
    PDS_obj = pr_calls_res,
    segment_method = "unique"
)
epi_padj_uniq <- calcEpitopePValues(
    probe_pds = pr_calls_res,
    epitope_ids = epi_segments_uniq_res,
    metap_method = "wilkinsons_max1"
)
prot_padj_uniq <- calcProteinPValues(
    epitope_ds = epi_padj_uniq,
    metap_method = "tippetts"
)
prot_calls <- makeProteinCalls(prot_padj_uniq)
```

---

min_max *Cap vector at minimum/maximum values*

Description

Cap vector at minimum/maximum values

Usage

```
min_max(val, min.value, max.value)
```

Arguments

val	vector of values to cap
min.value	minimum value
max.value	maximum value

Value

vector of capped values

Examples

```
min_max(10, 1, 5)
```

oneHitEpitopes *Find One-hit epitopes*

Description

Find One-hit epitopes

Usage

```
oneHitEpitopes(sample_epitopes)
```

Arguments

sample_epitopes	logical epitope matrix from makeCalls

Value

vector of one-hit, one-probe epitopes

Examples

```
hit_mat = data.frame(
row.names = c("A_1_1","A_2_2","A_3_3","A_4_4"),
sample1 = c(TRUE, FALSE, FALSE, TRUE),
sample2 = c(TRUE, TRUE, FALSE, FALSE),
sample3 = c(TRUE, TRUE, FALSE, FALSE)
)
oneHitEpitopes(hit_mat)
```

oneHitProbes *Find one hit probes*

Description

Find one hit probes

Usage

```
oneHitProbes(sample_probes)
```

Arguments

sample_probes	logical probe matrix from makeCalls

Value

vector of probes that are one-hits

Examples

```
hit_mat <- data.frame(
row.names = c("A;1","A;2","A;3","A;4"),
sample1 = c(TRUE, FALSE, FALSE, TRUE),
sample2 = c(TRUE, TRUE, FALSE, FALSE),
sample3 = c(TRUE, TRUE, FALSE, FALSE)
)
oneHitProbes(hit_mat)
```

---

oneProbeEpitopes *Indicate which epitopes are just one probe.*

Description

Indicate which epitopes are just one probe.

Usage

```
oneProbeEpitopes(epitope_ids)
```

Arguments

epitope_ids	vector of epitope ids

Value

vector of logical indicating epitopes that are one probe

Examples

```
oneProbeEpitopes(c("A_1_1", "B_1_1","C_1_2"))
```

---

probeHitSupported *Find probe hits with a consecutive probe or another sample*

Description

Find probe hits with a consecutive probe or another sample

Usage

```
probeHitSupported(hit_mat)
```

Arguments

hit_mat	matrix of logical values that indicate a hit with a TRUE value

Value

matrix of logical values indicate that the TRUE hit is supported by a consecutive probe hit in the sample sample or the within another sample

---

pvalue_to_zscore *Convert p-value matrix to a z-score matrix*

Description

Convert p-value matrix to a z-score matrix

Usage

```
pvalue_to_zscore(mat.in, one.sided = TRUE, log.p = FALSE, inf.zscore = 16)
```

Arguments

mat.in	matrix of p-values
one.sided	p-values one-sided
log.p	are p-values log transformed?
inf.zscore	infinite z-scores are capped to this value

Value

matrix of z-scores

Examples

```
mat <- matrix(runif(100), nrow=10)
rownames(mat) <- paste0("A;",seq_len(nrow(mat)))
pvalue_to_zscore(mat)
```

---

quantileNormalize *Normalize the exprs assay using quantile normalization*

Description

Normalize the exprs assay using quantile normalization

Usage

```
quantileNormalize(se)
```

Arguments

se	SummarizedExperiment with exprs assay

Value

SummarizedExperiment with exprs assay normalized

Examples

```
data(heffron2021_wuhan)
seq_ds_qn <- quantileNormalize(heffron2021_wuhan)
```

`smoothProbeDS` *Smooth probes across protein tiling*

Description

Smooth probes across protein tiling

Usage

```
smoothProbeDS(probe_ds, w = 2, eps = 1e-06)
```

Arguments

`probe_ds`	HERONProbeDataSet to smooth
`w`	smoothing width, probes +/- w/2 before and after are used
`eps`	error tolerance

Value

HERONProbeDataSet with smoothed data in exprs object

Examples

```
data(heffron2021_wuhan)
probe_ds <- convertSequenceDSToProbeDS(heffron2021_wuhan)
smoothed_ds <- smoothProbeDS(probe_ds)
```

Index

∗ **datasets**
heffron2021_wuhan, 18
∗ **internal**
HERON-package, 3
.HERONEpitopeDataSet
*(*HERONEpitopeDataSet-class*)*, 19
.HERONProbeDataSet
*(*HERONProbeDataSet-class*)*, 20
.HERONProteinDataSet
*(*HERONProteinDataSet-class*)*, 20
.HERONSequenceDataSet
*(*HERONSequenceDataSet-class*)*, 21

addSequenceAnnotations, 3

calcCombPValues, 4
calcEpitopePValues, 5
calcProbePValuesTPaired, 6
calcProbePValuesTUnpaired, 7
calcProbePValuesWUnpaired, 8
calcProteinPValues, 8
catSequences, 9
convertSequenceDSToProbeDS, 10

findBlocksProbeT, 11
findBlocksT, 11
findEpitopeSegments, 12

getEpitopeID, 13
getEpitopeIDsToProbeIDs, 13
getEpitopeProbeIDs, 14
getEpitopeProtein, 15
getEpitopeStart, 15
getEpitopeStop, 16
getKofN, 16
getProteinLabel, 17
getProteinStart, 17
getProteinTiling, 18

heffron2021_wuhan, 18
HERON *(*HERON-package*)*, 3
HERON-package, 3
HERONEpitopeDataSet
*(*HERONEpitopeDataSet-class*)*, 19
HERONEpitopeDataSet-class, 19
HERONProbeDataSet
*(*HERONProbeDataSet-class*)*, 20
HERONProbeDataSet-class, 20
HERONProteinDataSet
*(*HERONProteinDataSet-class*)*, 20
HERONProteinDataSet-class, 20
HERONSequenceDataSet
*(*HERONSequenceDataSet-class*)*, 21
HERONSequenceDataSet-class, 21

log2Transform, 21

makeEpitopeCalls, 22
makeProbeCalls, 23
makeProteinCalls, 23
min_max, 24

oneHitEpitopes, 25
oneHitProbes, 25
oneProbeEpitopes, 26

probeHitSupported, 26
pvalue_to_zscore, 27

quantileNormalize, 27

smoothProbeDS, 28