Package 'PhIPData'

March 26, 2024

Type Package

Title Container for PhIP-Seq Experiments

Version 1.10.0

Description PhIPData defines an S4 class for phage-immunoprecipitation sequencing (PhIP-seq) experiments. Buliding upon the RangedSummarizedExperiment class, PhIPData enables users to coordinate metadata with experimental data in analyses. Additionally, PhIPData provides specialized methods to subset and identify beads-only samples, subset objects using virus aliases, and use existing peptide libraries to populate object parameters.

License MIT + file LICENSE

Encoding UTF-8

LazyData false

Depends R (>= 4.1.0), SummarizedExperiment (>= 1.3.81)

Imports BiocFileCache, BiocGenerics, methods, GenomicRanges, IRanges, S4Vectors, edgeR, cli, utils

Suggests BiocStyle, testthat, knitr, rmarkdown, covr, dplyr, readr, withr

biocViews Infrastructure, DataRepresentation, Sequencing, Coverage

BugReports https://github.com/athchen/PhIPData/issues

Collate 'defineBeads.R' 'PhIPData-class.R' 'alias.R' 'library.R' 'subset.R' 'summaries.R' 'zzz.R'

RoxygenNote 7.1.2

VignetteBuilder knitr

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

git_branch RELEASE_3_18

git_last_commit 463fdc6

git_last_commit_date 2023-10-24

Repository Bioconductor 3.18

Date/Publication 2024-03-25

Author Athena Chen [aut, cre] (<https://orcid.org/0000-0001-6900-2264>),
Rob Scharpf [aut],
Ingo Ruczinski [aut]

Maintainer Athena Chen <achen70@jhu.edu>

R topics documented:

`aliases` *Using aliases to subset virus data*

Description

Rather than typing out full viruses names or repeating regexpressions, users can use aliases as a convenient tool to subset `PhIPData` objects by viral species.

Usage

```
getAlias(virus)

setAlias(virus, pattern)

deleteAlias(virus)
```

Arguments

`virus`	character vector of the alias
`pattern`	character vector of regexpressions corresponding to the alias

Details

Aliases are cached to an rda file containing only a `data.frame` with two columns: `alias` and `pattern`. The `alias` column contains the alias while the `pattern` column contains the corresponding regexpression of interest.

Once an alias is added to the database, it can always be accessed once the package is loaded. It is recommended to use the functions `setAlias` and `deleteAlias`. If an alias already exists in the database, `setAlias` replaces the matched pattern. If an alias does not exist in the database, `getAlias` returns `NA_character_`.

Value

getAlias() returns a vector of regexpressions corresponding to queried inputs. The returned vector is the same length as the input vector. Queries that do not exist in the database return NA_character_.

Functions

- getAlias: return a regexpression corresponding to the alias.
- setAlias: define/modify the regexpression for an alias.
- deleteAlias: remove an alias from the database.

Examples

```
## Edit and modify aliases in the database
setAlias("test_virus", "test_pattern")
getAlias("test_virus")
setAlias("test_virus", "test_pattern2")
getAlias("test_virus")
deleteAlias("test_virus")

## Edit and modify multiple aliases at once.
setAlias(c("virus_1", "virus_2"), c("pattern_1", "pattern_2"))
getAlias(c("virus_1", "virus_2"))
deleteAlias(c("virus_1", "virus_2"))

## Example of how to subset HIV using `getAlias`
## Often, it is useful to set the `ignore.case` of `grep`/`grepl` to TRUE.
counts_dat <- matrix(1:10, nrow = 5)
peptide_meta <- data.frame(species = c(
    rep("Epstein-Barr virus", 2),
    rep("human immunodeficiency virus", 3)
))

phip_obj <- PhIPData(counts = counts_dat, peptideInfo = peptide_meta)
subset(phip_obj, grepl(getAlias("HIV"), species, ignore.case = TRUE))
```

---

defineBeads *Defining how beads-only samples are encoded.*

Description

getBeadsName and setBeadsName are two function to get and set the string that encodes which samples are beads-only samples. Information about beads-only samples are stored in the groups column of sampleInfo.

Usage

```
getBeadsName()

setBeadsName(name)
```

Arguments

name	a string indicating how beads-only samples are encoded.

Details

If name is of length greater than one, only the first element of the vector is used. Non-character values of name are first coerced into strings.

Value

a string indicating how beads-only samples are encoded.

Functions

- getBeadsName: function that returns a string corresponding to how beads-only samples are encoded.
- setBeadsName: function to set the string that indicates which samples are beads-only samples in the groups column of sampleInfo.

Examples

```
## Returns the default string, "beads"
getBeadsName()

## Not run since it changes defaults/user settings
## Not run:
setBeadsName("beads-only")

## End(Not run)
```

---

librarySize *Calculate total read counts for each sample.*

Description

This function is a wrapper function for colSums on the counts assay.

Usage

```
librarySize(object, ..., withDimnames = TRUE)
```

Arguments

object	PhIPData object
...	arguments passed to colSums
withDimnames	logical; if true, the vector names are the sample names; otherwise the vector is unnamed.

Value

a (named) numeric vector. The length of the vector is equal to the number of samples.

Examples

```
example("PhIPData")
librarySize(phip_obj)

## Return an unnamed vector
librarySize(phip_obj, withDimnames = FALSE)
```

`peptideLibraries` *Peptide libraries*

Description

PhIP-Seq experiments often use identical peptide libraries different cohorts. These functions enable the user to conveniently reuse tidied libraries.

Usage

```
getLibrary(name)

makeLibrary(library, name)

removeLibrary(name)

listLibrary()
```

Arguments

`name`	name of the library
`library`	a `matrix`, `data.frame`, or DataFrame with the peptide information for the specified library.

Details

Each library is stored as a DataFrame in .rds file. New libraries can be stored for future use with the `makeLibrary` function.

Value

`getLibrary` returns a DataFrame corresponding to the peptide information for the specified library.

Functions

- `getLibrary`: return a DataFrame with the peptide information corresponding to the library.
- `makeLibrary`: create and store a DataFrame with the specified peptide information.
- `removeLibrary`: delete stored libraries
- `listLibrary`: list all available libraries

Examples

```
## Create a new library
pep_meta <- data.frame(species = c(
    rep("human immunodeficiency virus", 3),
    rep("Epstein-Barr virus", 2)
))
makeLibrary(pep_meta, "new_library")

## Use new library
counts_dat <- matrix(1:10, nrow = 5)
phip_obj <- PhIPData(
    counts = counts_dat,
    peptideInfo = getLibrary("new_library")
)

## List libraries
listLibrary()

## Delete created library
removeLibrary("new_library")
```

PhIPData-class *The PhIPData class*

Description

The `PhIPData` class is a matrix-like container designed to organize results from phage-immunoprecipitation (PhIP-Seq) experiments. Rows in PhIPData objects represent peptides and columns represent samples. Each object contains at least three assays:

- `counts`: a matrix of raw read counts,
- `logfc`: a matrix of log2 estimated fold-change in comparison to beads-only samples,
- `prob`: a matrix of probabilities associated with whether a sample has an enriched antibody response for a peptide.

The `PhIPData` class extends the RangedSummarizedExperiment class, so methods documented in RangedSummarizedExperiment and SummarizedExperiment also work on `PhIPData` objects.

Usage

```
PhIPData(
  counts = matrix(nrow = 0, ncol = 0),
  logfc = matrix(nrow = 0, ncol = 0),
  prob = matrix(nrow = 0, ncol = 0),
  peptideInfo = S4Vectors::DataFrame(),
  sampleInfo = S4Vectors::DataFrame(),
  metadata = list(),
  .defaultNames = "info"
)
```

Arguments

counts	a matrix, data.frame, or DataFrameof **integer** read counts.
logfc	a matrix, data.frame, or DataFrame of log2 estimated fold changes.
prob	a matrix, data.frame, or DataFrame of probability values (p-values or posterior probabilities) for enrichment estimates.
peptideInfo	a data.frame or DataFrame of peptide information.
sampleInfo	a data.frame or DataFrame of additional sample information.
metadata	a list object containing experiment-specific metadata.
.defaultNames	vector of names to use when sample and peptide identifiers disagree across the metadata and the counts, logfc, and prob matrices. If .defaultNames is of length 1, the same source is used for both peptide and sample identifiers. If .defaultNames is longer than 2, the first and second elements correspond to the names for peptides and samples, respectively. Valid options are: • "info": names should be taken from the SampleInfo or peptideInfo objects. • "counts": names should be taken from the row/column names of the counts object. • "logfc": names should be taken from the row/column names of the logfc object. • "prob": names should be taken from the row/column names of the prob object.

Details

Rows of PhIPData objects correspond to peptides of interest and are organized in GRanges or GRangesList objects. Though originally designed for genomic ranges, the sequence name and genomic range information in GRanges objects can be replaced with peptide names and amino acid positions, respectively. If no peptide names are given, peptides are given the names of pep_rownum. Peptide positions are specified by columns pos_start and pos_end in the peptideInfo argument of the constuctor. Missing position information is set to 0. Additional peptide annotation can also be stored in GRanges objects and can be used to subset PhIPData objects as shown below.

Columns of PhIPData objects represent samples. Sample metadata are stored in a DataFrame and can be accessed as shown below. If no sample names are specified, samples are given default names of sample_colnum.

Unlike RangedSummarizedExperiment/SummarizedExperiment objects, PhIPData objects must contain counts, logfc, prob. If any of the three assays are missing when the constructor is called, an empty matrix of the same names and dimensions is initialized for that assay. Sample and peptide names are harmonized across assays and annotation during construction and replacement.

Though 'counts' typically contain integer values for the number of reads aligned to each peptide, 'PhIPData' only requires that stored values are non-negative numeric values. Pseudocounts or non-integer count values can also be stored in the 'counts' assay.

Value

A PhIPData object.

Constructor

PhIPData objects are constructed using the homonymous function and arguments as described above. Any PhIPData object can be created so long as peptide and sample identifiers (or lack thereof) are specified via any of the parameters.

See Also

PhIPData-methods for accessors and modifiers for PhIPData components. SummarizedExperiment

Examples

```
## Construct a new PhIPData object
counts_dat <- matrix(sample(1:1e6, 25, replace = TRUE), nrow = 5)
logfc_dat <- matrix(rnorm(25, 0, 10), nrow = 5)
prob_dat <- matrix(rbeta(25, 1, 1), nrow = 5)

peptide_meta <- data.frame(
    pos_start = 1:5,
    pos_end = 6:10,
    species = c(rep("HIV", 3), rep("EBV", 2))
)
sample_meta <- data.frame(
    gender = sample(c("M", "F"), 5, TRUE),
    group = sample(c("ctrl", "trt", "beads"), 5, TRUE)
)
exp_meta <- list(
    date_run = as.Date("2021/01/20"),
    reads_per_sample = colSums(counts_dat)
)

rownames(counts_dat) <- rownames(logfc_dat) <-
    rownames(prob_dat) <- rownames(peptide_meta) <-
    paste0("pep_", 1:5)
colnames(counts_dat) <- colnames(logfc_dat) <-
```

```
    colnames(prob_dat) <- rownames(sample_meta) <-
    paste0("sample_", 1:5)

phip_obj <- PhIPData(
    counts_dat, logfc_dat, prob_dat,
    peptide_meta, sample_meta, exp_meta
)
phip_obj
```

PhIPData-methods *Accessing and Modifying Information in PhIPData objects*

Description

Methods to extract and modify assay(s)(including convenient functions for counts, logfc, and prob), sampleInfo, peptideInfo, and metadata.

Usage

```
## S4 method for signature 'PhIPData'
counts(object, ...)

logfc(object, ...)

## S4 method for signature 'PhIPData'
logfc(object, ...)

prob(object, ...)

## S4 method for signature 'PhIPData'
prob(object, ...)

peptideInfo(object, ...)

## S4 method for signature 'PhIPData'
peptideInfo(object, ...)

sampleInfo(object, ...)

## S4 method for signature 'PhIPData'
sampleInfo(object, ...)

## S4 replacement method for signature 'PhIPData,list'
assays(x, withDimnames = TRUE, ...) <- value

## S4 replacement method for signature 'PhIPData,SimpleList'
assays(x, withDimnames = TRUE, ...) <- value
```

```
## S4 replacement method for signature 'PhIPData,missing'
assay(x, i, withDimnames = TRUE, ...) <- value

## S4 replacement method for signature 'PhIPData,numeric'
assay(x, i, withDimnames = TRUE, ...) <- value

## S4 replacement method for signature 'PhIPData,character'
assay(x, i, withDimnames = TRUE, ...) <- value

## S4 replacement method for signature 'PhIPData'
counts(object, ...) <- value

logfc(object, ...) <- value

## S4 replacement method for signature 'PhIPData'
logfc(object, ...) <- value

prob(object, ...) <- value

## S4 replacement method for signature 'PhIPData'
prob(object, ...) <- value

peptideInfo(object) <- value

## S4 replacement method for signature 'PhIPData'
peptideInfo(object) <- value

sampleInfo(object, ...) <- value

## S4 replacement method for signature 'PhIPData'
sampleInfo(object) <- value
```

Arguments

object	A PhIPData object
...	parameters for assays, which are typically not needed.
x	A PhIPData object
withDimnames	Parameter for RangedSummarizedExperiment class functions. Overrided since row/column names are automatically synced within each object.
value	A matrix, data.frame, or DataFrame of the same dimensions (not necessarily the same names)
i	A numeric, character

Details

In addition to the functions detailed in RangedSummarizedExperiment, the PhIPData class includes conveniently named functions to quickly access and modify frequently used components of PhIPData objects.

Replacement functions ensure that names of the replacement object are matched with the names of the PhIPData object.

Since packages for identifying differential expression in RNA-seq experiments are frequently used for estimating fold-changes for peptide enrichments, the class also includes coercion methods to and from DGELists.

Value

Accessors: a DataFrame object

Setters: a PhIPData object

Available methods

In the following code snippets, x is a PhIPData object, value is a matrix-like object with the same dimensions as x, and ... are further arguments passed to assay (for the getter) or assay<- (for the setter).

counts(x, ...), counts(x, ...) <- value: Get or set a matrix of raw read counts

logfc(x, ...), logfc(x, ...) <- value: Get or set a matrix of log2 estimated fold changes (in comparison to beads-only samples)

prob(x, ...), pob(x, ...) <- value: Get or set a matrix of probabilities associated with whether a sample has an enriched antibody response for a peptide.

See Also

assays for SummarizedExperiment operations.

Examples

```
example("PhIPData")

replacement_dat <- matrix(1L, nrow = 5, ncol = 5)

## SummarizedExperiment Accessors and Setters
assays(phip_obj)
assays(phip_obj)$counts <- replacement_dat
assay(phip_obj, "logfc")
assay(phip_obj, "logfc") <- replacement_dat

## counts
counts(phip_obj)
counts(phip_obj) <- counts_dat

## logfc
logfc(phip_obj)
logfc(phip_obj) <- logfc_dat

## prob
prob(phip_obj)
prob(phip_obj) <- replacement_dat
```

```
## coercion functions
as(phip_obj, "DGEList")
as(phip_obj, "List")
as(phip_obj, "list")
```

`propReads` *Proportion of sample reads*

Description

This function calculates the proportion of total sample reads pulled by each peptide.

Usage

```
propReads(object, withDimnames = TRUE)
```

Arguments

`object`	PhIPData object
`withDimnames`	logical; if true return a matrix with the same dimension names as the original object.

Value

A (named) numeric matrix with the same dimensions as the function input. Matrix values are between 0 and 1.

Examples

```
example("PhIPData")
propReads(phip_obj)

## Return an unnamed matrix
propReads(phip_obj, withDimnames = FALSE)
```

`subsetBeads` *Subset beads-only samples*

Description

Function to subset PhIP-seq data for beads-only samples.

Usage

```
subsetBeads(object)
```

Arguments

`object`	PhIPData object

Value

a PhIPData object.

Examples

```
example("PhIPData")
subsetBeads(phip_obj)
```

Index