Package ‘OmicNavigator’

August 5, 2021

Type Package

Title Open-Source Software for ‘Omic’ Data Analysis and Visualization

Description A tool for interactive exploration of the results from ‘omics’ experiments to facilitate novel discoveries from high-throughput biology. The software includes R functions for the ‘bioinformatician’ to deposit study metadata and the outputs from statistical analyses (e.g. differential expression, enrichment). These results are then exported to an interactive JavaScript dashboard that can be interrogated on the user’s local machine or deployed online to be explored by collaborators. The dashboard includes ‘sortable’ tables, interactive plots including network visualization, and fine-grained filtering based on statistical significance.

Version 1.4.3

URL https://github.com/abbvie-external/OmicNavigator

BugReports https://github.com/abbvie-external/OmicNavigator/issues

License MIT + file LICENSE

License_restricts_use no

License_is_FOSS yes

Encoding UTF-8

LazyData true

Depends R (>= 3.2.0)

Imports data.table (>= 1.12.4), graphics, jsonlite, stats, tools, utils

Suggests faviconPlease, ggplot2, opencpu, tinytest (>= 1.2.3), ttdo (>= 0.0.6), UpSetR

RoxygenNote 7.1.1

NeedsCompilation no

Author Terrence Ernst [aut] (Web application), John Blischak [aut, cre] (<https://orcid.org/0000-0003-2634-9879>), Paul Nordlund [aut] (Web application), Justin Moore [aut] (UpSet-related functions and web application),
Joe Dalen [aut] (Barcode functionality and web application),
Akshay Bhamidipati [aut] (Web application),
Brett Engelmann [aut],
AbbVie Inc. [cph, fnd]

Maintainer John Blischak <jdblischak@gmail.com>

Repository CRAN

Date/Publication 2021-08-05 07:20:02 UTC

R topics documented:

addAnnotations .................................................. 3
addAssays ......................................................... 4
addBarcodes ....................................................... 5
addEnrichments .................................................... 6
addEnrichmentsLinkouts ........................................... 6
addFeatures ....................................................... 8
addMetaFeatures ............................................... 8
addMetaFeaturesLinkouts ......................................... 9
addModels ......................................................... 10
addOverlaps ....................................................... 11
addPlots .......................................................... 12
addReports ......................................................... 13
addResults ......................................................... 14
addResultsLinkouts ............................................... 14
addSamples ....................................................... 16
addTests .......................................................... 16
basal.vs.lp ....................................................... 18
basal.vs.ml ....................................................... 19
cam.BasalvsLP ................................................... 20
cam.BasalvsML ................................................... 21
createStudy ...................................................... 22
exportStudy ...................................................... 25
getAnnotations .................................................. 26
getAssays ........................................................ 27
getBarcodeData .................................................. 28
getBarcodes ....................................................... 29
gerEnrichments .................................................. 29
gerEnrichmentsIntersection ..................................... 30
gerEnrichmentsLinkouts ......................................... 31
gerEnrichmentsNetwork .......................................... 32
gerEnrichmentsTable ............................................ 33
gerEnrichmentsUpset ............................................ 34
gerFavicons ....................................................... 34
gerFeatures ....................................................... 35
gerInstalledStudies .............................................. 36
gerLinkFeatures .................................................. 36
gerMetaFeatures .................................................. 37
addAnnotations

Description
Add annotations

Usage
addAnnotations(study, annotations, reset = FALSE)
**Arguments**

- **study**: An OmicNavigator study created with `createStudy`
- **annotations**: The annotations used for the enrichment analyses. The input is a nested list. The top-level list contains one entry per annotation database, e.g. reactome. The names correspond to the name of each annotation database. Each of these elements should be list of that contains more information about each annotation database. Specifically the sub-list should contain 1) description, a character vector that describes the resource, 2) featureID, the name of the column in the features table that was used for the enrichment analysis, and 3) terms, a list of annotation terms. The names of terms sublist correspond to the name of the annotation terms. Each of the annotation terms should be a character vector of featureIDs.
- **reset**: Reset the data prior to adding the new data (default: FALSE). The default is to add to or modify any previously added data (if it exists). Setting `reset = TRUE` enables you to remove existing data you no longer want to include in the study.

**Value**

Returns the original `onStudy` object passed to the argument `study`, but modified to include the newly added data.

---

**Description**

Add assays

**Usage**

`addAssays(study, assays, reset = FALSE)`

**Arguments**

- **study**: An OmicNavigator study created with `createStudy`
- **assays**: The assays from the study. The input object is a list of data frames (one per model). The row names should correspond to the featureIDs (`addFeatures`). The column names should correspond to the sampleIDs (`addSamples`). The data frame should only contain numeric values. To share a data frame across multiple models, use the modelID "default".
- **reset**: Reset the data prior to adding the new data (default: FALSE). The default is to add to or modify any previously added data (if it exists). Setting `reset = TRUE` enables you to remove existing data you no longer want to include in the study.
addBarcodes

**Description**

The app can display a barcode plot of the enrichment results for a given annotation term. The metadata in 'barcodes' instructs the app how to create and label the barcode plot.

**Usage**

```r
addBarcodes(study, barcodes, reset = FALSE)
```

**Arguments**

- **study**: An OmicNavigator study created with `createStudy`
- **barcodes**: The metadata variables that describe the barcode plot. The input object is a list of lists (one per model). Each sublist must contain the element **statistic**, which is the column name in the results table to use to construct the barcode plot. Each sublist may additionally contain any of the following optional elements: 1) absolute - Should the statistic be converted to its absolute value (default is `TRUE`). 2) logFoldChange - The column name in the results table that contains the log fold change values. 3) labelStat - The x-axis label to describe the statistic. 4) labelLow - The left-side label to describe low values of the statistic. 5) labelHigh - The right-side label to describe high values of the statistic. 6) featureDisplay - The feature variable to use to label the barcode plot on hover. To share metadata across multiple models, use the modelID "default".
- **reset**: Reset the data prior to adding the new data (default: `FALSE`). The default is to add to or modify any previously added data (if it exists). Setting `reset = TRUE` enables you to remove existing data you no longer want to include in the study.

**Value**

Returns the original `onStudy` object passed to the argument `study`, but modified to include the newly added data.
addEnrichments

Add enrichment results

Usage

addEnrichments(study, enrichments, reset = FALSE)

Arguments

study An OmicNavigator study created with createStudy

enrichments The enrichment results from each model. The input is a nested named list. The names of the list correspond to the model names. Each list element should be a list of the annotation databases tested (addAnnotations). The names of the list correspond to the annotation databases. Each list element should be another list of tests (addTests). The names correspond to the tests performed. Each of these elements should be a data frame with enrichment results. Each table must contain the following columns: "termID", "description", "nominal" (the nominal statistics), and "adjusted" (the statistics after adjusting for multiple testing). Any additional columns are ignored.

reset Reset the data prior to adding the new data (default: FALSE). The default is to add to or modify any previously added data (if it exists). Setting reset = TRUE enables you to remove existing data you no longer want to include in the study.

Value

Returns the original onStudy object passed to the argument study, but modified to include the newly added data

addEnrichmentsLinkouts

Add linkouts to external resources in the enrichments table

Description

You can provide additional information on the annotation terms in your study by providing linkouts to external resources. These will be embedded directly in the enrichments table.

Usage

addEnrichmentsLinkouts(study, enrichmentsLinkouts, reset = FALSE)
addEnrichmentsLinkouts

Arguments

study An OmicNavigator study created with createStudy

enrichmentsLinkouts The URL patterns that describe linkouts to external resources (see Details below). The input object is a named list. The names of the list correspond to the annotation names. Each element of the list is a character vector of linkouts for that annotationID.

reset Reset the data prior to adding the new data (default: FALSE). The default is to add to or modify any previously added data (if it exists). Setting reset = TRUE enables you to remove existing data you no longer want to include in the study.

Details

For each linkout, the URL pattern you provide will be concatenated with the value of the termID column. As an example, if you used the annotation database AmiGO 2 for your enrichments analysis, you can provide a linkout for each termID using the following pattern:

\[ \text{go} = \text{"http://amigo.geneontology.org/amigo/term/"} \]

As another example, if you used the annotation database Reactome for your enrichments analysis, you can provide a linkout for each termID using the following pattern:

\[ \text{reactome} = \text{"https://reactome.org/content/detail/"} \]

Note that you can provide more than one linkout per termID.

Value

Returns the original onStudy object passed to the argument study, but modified to include the newly added data

See Also

addAnnotations, addEnrichments

Examples

```r
study <- createStudy("example")
enrichmentsLinkouts <- list(
  gobp = c("http://amigo.geneontology.org/amigo/term/",
           "https://www.ebi.ac.uk/QuickGO/term/"),
  reactome = "https://reactome.org/content/detail/"
)
study <- addEnrichmentsLinkouts(study, enrichmentsLinkouts)
```
addFeatures (Add feature metadata)

Usage

addFeatures(study, features, reset = FALSE)

Arguments

study An OmicNavigator study created with createStudy
features The metadata variables that describe the features in the study. The input object is a list of data frames (one per model). The first column of each data frame is used as the featureID, so it must contain unique values. To share a data frame across multiple models, use the modelID "default". All columns will be coerced to character strings.
reset Reset the data prior to adding the new data (default: FALSE). The default is to add to or modify any previously added data (if it exists). Setting reset = TRUE enables you to remove existing data you no longer want to include in the study.

Value

Returns the original onStudy object passed to the argument study, but modified to include the newly added data.

addMetaFeatures (Add meta-feature metadata)

Description

The meta-features table is useful anytime there are metadata variables that cannot be mapped 1:1 to your features. For example, a peptide may be associated with multiple proteins.

Usage

addMetaFeatures(study, metaFeatures, reset = FALSE)
Arguments

study An OmicNavigator study created with createStudy

metaFeatures The metadata variables that describe the meta-features in the study. The input object is a list of data frames (one per model). The first column of each data frame is used as the featureID, so it must contain the same IDs as the corresponding features data frame (addFeatures). To share a data frame across multiple models, use the modelID "default". All columns will be coerced to character strings.

reset Reset the data prior to adding the new data (default: FALSE). The default is to add to or modify any previously added data (if it exists). Setting reset = TRUE enables you to remove existing data you no longer want to include in the study.

Value

Returns the original onStudy object passed to the argument study, but modified to include the newly added data

Description

You can provide additional information on the metaFeatures in your study by providing linkouts to external resources. These will be embedded directly in the metaFeatures table.

Usage

addMetaFeaturesLinkouts(study, metaFeaturesLinkouts, reset = FALSE)

Arguments

study An OmicNavigator study created with createStudy

metaFeaturesLinkouts The URL patterns that describe linkouts to external resources (see Details below). The input object is a nested named list. The names of the list correspond to the model names. Each element of the list is a named list of character vectors. The names of this nested list must correspond to the column names of the matching metaFeatures table (addMetaFeatures). To share linkouts across multiple models, use the modelID "default".

reset Reset the data prior to adding the new data (default: FALSE). The default is to add to or modify any previously added data (if it exists). Setting reset = TRUE enables you to remove existing data you no longer want to include in the study.
Details

For each linkout, the URL pattern you provide will be concatenated with the value of that column for each row. As an example, if your metaFeatures table included a column named "ensembl" that contained the Ensembl Gene ID for each feature, you could create a linkout to Ensembl using the following pattern:

`ensembl = "https://ensembl.org/Homo_sapiens/Gene/Summary?g="`

As another example, if you had a column named "entrez" that contained the Entrez Gene ID for each feature, you could create a linkout to Entrez using the following pattern:


Note that you can provide more than one linkout per column.

Value

Returns the original onStudy object passed to the argument study, but modified to include the newly added data.

See Also

`addMetaFeatures`

Examples

```r
study <- createStudy("example")
metaFeaturesLinkouts <- list(
  default = list(
    ensembl = c("https://ensembl.org/Homo_sapiens/Gene/Summary?g=",
                 "https://www.genome.ucsc.edu/cgi-bin/hgGene?hgg_gene="),
  )
)
study <- addMetaFeaturesLinkouts(study, metaFeaturesLinkouts)
```

Add models

Usage

`addModels(study, models, reset = FALSE)`
addOverlaps

Arguments

- **study**: An OmicNavigator study created with `createStudy`
- **models**: The models analyzed in the study. The input is a named list. The names correspond to the names of the models. The elements correspond to the descriptions of the models. Alternatively, instead of a single character string, you can provide a list of metadata fields about each model. The field "description" will be used to derive the tooltip displayed in the app.
- **reset**: Reset the data prior to adding the new data (default: FALSE). The default is to add to or modify any previously added data (if it exists). Setting `reset = TRUE` enables you to remove existing data you no longer want to include in the study.

Value

Returns the original `onStudy` object passed to the argument `study`, but modified to include the newly added data.

Examples

```r
study <- createStudy("example")
models <- list(
  model_01 = "Name of first model",
  model_02 = "Name of second model"
)
study <- addModels(study, models)

# Alternative: provide additional metadata about each model
models <- list(
  model_01 = list(
    description = "Name of first model",
    data_type = "transcriptomics"
  ),
  model_02 = list(
    description = "Name of second model",
    data_type = "proteomics"
  )
)
```

---

## addOverlaps

*Add overlaps between annotation gene sets*

Description

The app’s network view of the enrichments results requires pairwise overlap metrics between all the terms of each annotation in order to draw the edges between the nodes/terms. These overlaps are calculated automatically when installing or exporting an OmicNavigator study. If you’d like, you can manually calculate these pairwise overlaps by calling `addOverlaps` prior to installing or exporting your study.
Usage

```r
addOverlaps(study, reset = FALSE)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>study</td>
<td>An OmicNavigator study created with <code>createStudy</code></td>
</tr>
<tr>
<td>reset</td>
<td>Reset the data prior to adding the new data (default: FALSE). The default is to add to or modify any previously added data (if it exists). Setting <code>reset = TRUE</code> enables you to remove existing data you no longer want to include in the study.</td>
</tr>
</tbody>
</table>

Value

Returns the original `onStudy` object passed to the argument `study`, but modified to include the newly added data.

---

**addPlots**

Add custom plotting functions

**Description**

Include custom plots that the app will display when a feature is selected by the user.

**Usage**

```r
addPlots(study, plots, reset = FALSE)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>study</td>
<td>An OmicNavigator study created with <code>createStudy</code></td>
</tr>
<tr>
<td>plots</td>
<td>Custom plotting functions for the study. The input object is a nested list. The first list corresponds to the modelID(s). The second list corresponds to the name(s) of the function(s) defined in the current R session. The third list provides metadata to describe each plot. The only required metadata element is <code>displayName</code>, which controls how the plot will be named in the app. You are encouraged to also specify the <code>plotType</code>, e.g. &quot;singleFeature&quot;, &quot;multiFeature&quot;. If you do not specify the <code>plotType</code>, the plot will be assumed to be &quot;singleFeature&quot;. Optionally, if the plotting function requires external packages, these can be defined in the element <code>packages</code>. To share plots across multiple models, use the modelID &quot;default&quot;.</td>
</tr>
<tr>
<td>reset</td>
<td>Reset the data prior to adding the new data (default: FALSE). The default is to add to or modify any previously added data (if it exists). Setting <code>reset = TRUE</code> enables you to remove existing data you no longer want to include in the study.</td>
</tr>
</tbody>
</table>
Details

Custom plotting functions are passed a list of data frames: assays with the measurements, features with the feature data, and samples with the sample data. Both assays and features are subset to only include data for the specified featureID(s) (and re-ordered so their rows match). Thus your custom plotting function must have at least one argument. It can have additional arguments if you wish, but these must be provided with default values, because plotStudy only passes the plotting data to the first argument.

Note that any ggplot2 plots will require extra care. This is because the plotting code will be inserted into a study package, and thus must follow the best practices for using ggplot2 within packages. Specifically, when you refer to columns of the data frame, e.g. `aes(x = group)`, you need to prefix it with `.data$`, so that it becomes `aes(x = .data$group)`. Fortunately this latter code will also run fine as you interactively develop the function.

Value

Returns the original onStudy object passed to the argument study, but modified to include the newly added data

See Also

getPlottingData, plotStudy

---

**Description**

You can include reports of the analyses you performed to generate the results.

**Usage**

```r
addReports(study, reports, reset = FALSE)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>study</td>
<td>An OmicNavigator study created with <code>createStudy</code></td>
</tr>
<tr>
<td>reports</td>
<td>The analysis report(s) that explain how the study results were generated. The input object is a list of character vectors (one per model). Each element should be either a URL or a path to a file on your computer. If it is a path to a file, this file will be included in the exported study package. To share a report across multiple models, use the modelID &quot;default&quot;.</td>
</tr>
<tr>
<td>reset</td>
<td>Reset the data prior to adding the new data (default: FALSE). The default is to add to or modify any previously added data (if it exists). Setting <code>reset = TRUE</code> enables you to remove existing data you no longer want to include in the study.</td>
</tr>
</tbody>
</table>
Value

Returns the original `onStudy` object passed to the argument `study`, but modified to include the newly added data.

---

### addResults

#### Description

Add inference results

#### Usage

```r
addResults(study, results, reset = FALSE)
```

#### Arguments

- **study**: An OmicNavigator study created with `createStudy`
- **results**: The inference results from each model. The input is a nested named list. The names of the list correspond to the model names. Each element in the list should be a list of data frames with inference results, one for each test. In each data frame, the `featureID` must be in the first column, and all other columns must be numeric.
- **reset**: Reset the data prior to adding the new data (default: `FALSE`). The default is to add to or modify any previously added data (if it exists). Setting `reset = TRUE` enables you to remove existing data you no longer want to include in the study.

#### Value

Returns the original `onStudy` object passed to the argument `study`, but modified to include the newly added data.

---

### addResultsLinkouts

#### Description

You can provide additional information on the features in your study by providing linkouts to external resources. These will be embedded directly in the results table.

#### Usage

```r
addResultsLinkouts(study, resultsLinkouts, reset = FALSE)
```
Arguments

study  An OmicNavigator study created with \texttt{createStudy}

resultsLinkouts  The URL patterns that describe linkouts to external resources (see Details below). The input object is a nested named list. The names of the list correspond to the model names. Each element of the list is a named list of character vectors. The names of this nested list must correspond to the column names of the matching features table. To share linkouts across multiple models, use the modelID "default".

reset  Reset the data prior to adding the new data (default: \texttt{FALSE}). The default is to add to or modify any previously added data (if it exists). Setting \texttt{reset = TRUE} enables you to remove existing data you no longer want to include in the study.

Details

For each linkout, the URL pattern you provide will be concatenated with the value of that column for each row. As an example, if your features table included a column named "ensembl" that contained the Ensembl Gene ID for each feature, you could create a linkout to Ensembl using the following pattern:

\[
\text{ensembl} = \text{"https://ensembl.org/Homo_sapiens/Gene/Summary?g="}
\]

As another example, if you had a column named "entrez" that contained the Entrez Gene ID for each feature, you could create a linkout to Entrez using the following pattern:

\[
\text{entrez} = \text{"https://www.ncbi.nlm.nih.gov/gene/"}
\]

Note that you can provide more than one linkout per column.

Value

Returns the original \texttt{onStudy} object passed to the argument \texttt{study}, but modified to include the newly added data.

See Also

\texttt{addFeatures}

Examples

```r
study <- createStudy("example")
resultsLinkouts <- list(
  default = list(
    ensembl = c("https://ensembl.org/Homo_sapiens/Gene/Summary?g=",
                 "https://www.genome.ucsc.edu/cgi-bin/hgGene?hgg_gene="),
  )
)
study <- addResultsLinkouts(study, resultsLinkouts)
```
addSamples

Add sample metadata

Usage

addSamples(study, samples, reset = FALSE)

Arguments

study An OmicNavigator study created with createStudy

samples The metadata variables that describe the samples in the study. The input object is a named list of data frames (one per model). The first column of each data frame is used as the sampleID, so it must contain unique values. To share a data frame across multiple models, use the modelID "default".

reset Reset the data prior to adding the new data (default: FALSE). The default is to add to or modify any previously added data (if it exists). Setting reset = TRUE enables you to remove existing data you no longer want to include in the study.

Value

Returns the original onStudy object passed to the argument study, but modified to include the newly added data

addTests

Add tests

Usage

addTests(study, tests, reset = FALSE)
addTests

Arguments

study An OmicNavigator study created with createStudy
tests The tests from the study. The input object is a list of lists. Each element of the top-level list is a model. The names should be the modelIDs. For each modelID, each element of the nested list is a test. The names should be the testIDs. The value should be a single character string describing the testID. To share tests across multiple models, use the modelID "default". Instead of a single character string, you can provide a list of metadata fields about each test. The field "description" will be used to derive the tooltip displayed in the app.
reset Reset the data prior to adding the new data (default: FALSE). The default is to add to or modify any previously added data (if it exists). Setting reset = TRUE enables you to remove existing data you no longer want to include in the study.

Value

Returns the original onStudy object passed to the argument study, but modified to include the newly added data.

Examples

```r
study <- createStudy("example")
tests <- list(
  default = list(
    test_01 = "Name of first test",
    test_02 = "Name of second test"
  )
)
study <- addTests(study, tests)

# Alternative: provide additional metadata about each test
tests <- list(
  default = list(
    test_01 = list(
      description = "Name of first test",
      comparison_type = "treatment vs control",
      effect_size = "beta"
    ),
    test_02 = list(
      description = "Name of second test",
      comparison_type = "treatment vs control",
      effect_size = "logFC"
    )
  )
)
```
Description

A subset of the object `basal.vs.lp` from Bioconductor workflow RNAseq123.

Usage

`basal.vs.lp`

Format

A data frame with 24 rows and 8 columns:

- **ENTREZID**: Entrez ID of mouse gene
- **SYMBOL**: Symbol of mouse gene
- **TXCHROM**: Chromosome location of mouse gene
- **logFC**: Log fold change
- **AveExpr**: Average expression level of the gene across all samples
- **t**: Moderated t-statistic
- **P.Value**: p-value
- **adj.P.Val**: Adjusted p-value

Source


References


Examples

```r
head(basal.vs.lp)
str(basal.vs.lp)
```
Description

A subset of the object `basal.vs.ml` from Bioconductor workflow RNAseq123.

Usage

`basal.vs.ml`

Format

A data frame with 24 rows and 8 columns:

- **ENTREZID**: Entrez ID of mouse gene
- **SYMBOL**: Symbol of mouse gene
- **TXCHROM**: Chromosome location of mouse gene
- **logFC**: Log fold change
- **AveExpr**: Average expression level of the gene across all samples
- **t**: Moderated t-statistic
- **P.Value**: p-value
- **adj.P.Val**: Adjusted p-value

Source


References


Examples

```r
head(basal.vs.ml)
str(basal.vs.ml)
```
Description

A subset of the object `cam.BasalvsLP` from Bioconductor workflow RNAseq123.

Usage

`cam.BasalvsLP`

Format

A data frame with 4 rows and 4 columns:

- **NGenes**: Number of genes in each term
- **Direction**: Direction of the enrichment
- **PValue**: Nominal p-value
- **FDR**: Multiple-testing adjusted p-value

Source


References


Examples

```r
head(cam.BasalvsLP)
str(cam.BasalvsLP)
```
Description

A subset of the object `cam.BasalvsML` from Bioconductor workflow RNAseq123.

Usage

`cam.BasalvsML`

Format

A data frame with 4 rows and 4 columns:

- **NGenes** Number of genes in each term
- **Direction** Direction of the enrichment
- **PValue** Nominal p-value
- **FDR** Multiple-testing adjusted p-value

Source


References


Examples

```r
head(cam.BasalvsML)
str(cam.BasalvsML)
```
createStudy

Create a new OmicNavigator study.

Usage

createStudy(
  name,
  description = name,
  samples = list(),
  features = list(),
  models = list(),
  assays = list(),
  tests = list(),
  annotations = list(),
  results = list(),
  enrichments = list(),
  metaFeatures = list(),
  plots = list(),
  barcodes = list(),
  reports = list(),
  resultsLinkouts = list(),
  enrichmentsLinkouts = list(),
  metaFeaturesLinkouts = list(),
  version = NULL,
  maintainer = NULL,
  maintainerEmail = NULL,
  studyMeta = list()
)

Arguments

name      Name of the study
description Description of the study
samples   The metadata variables that describe the samples in the study. The input object
          is a named list of data frames (one per model). The first column of each data
          frame is used as the sampleID, so it must contain unique values. To share a data
          frame across multiple models, use the modelID "default".
features  The metadata variables that describe the features in the study. The input object
          is a list of data frames (one per model). The first column of each data frame is
          used as the featureID, so it must contain unique values. To share a data frame
          across multiple models, use the modelID "default". All columns will be coerced
          to character strings.
models

The models analyzed in the study. The input is a named list. The names correspond to the names of the models. The elements correspond to the descriptions of the models. Alternatively, instead of a single character string, you can provide a list of metadata fields about each model. The field "description" will be used to derive the tooltip displayed in the app.

assays

The assays from the study. The input object is a list of data frames (one per model). The row names should correspond to the featureIDs (addFeatures). The column names should correspond to the sampleIDs (addSamples). The data frame should only contain numeric values. To share a data frame across multiple models, use the modelID "default".

tests

The tests from the study. The input object is a list of lists. Each element of the top-level list is a model. The names should be the modelIDs. For each modelID, each element of the nested list is a test. The names should be the testIDs. The value should be a single character string describing the testID. To share tests across multiple models, use the modelID "default". Instead of a single character string, you can provide a list of metadata fields about each test. The field "description" will be used to derive the tooltip displayed in the app.

annotations

The annotations used for the enrichment analyses. The input is a nested list. The top-level list contains one entry per annotation database, e.g. reactome. The names correspond to the name of each annotation database. Each of these elements should be list of that contains more information about each annotation database. Specifically the sublist should contain 1) description, a character vector that describes the resource, 2) featureID, the name of the column in the features table that was used for the enrichment analysis, and 3) terms, a list of annotation terms. The names of terms sublist correspond to the name of the annotation terms. Each of the annotation terms should be a character vector of featureIDs.

results

The inference results from each model. The input is a nested named list. The names of the list correspond to the model names. Each element in the list should be a list of data frames with inference results, one for each test. In each data frame, the featureID must be in the first column, and all other columns must be numeric.

enrichments

The enrichment results from each model. The input is a nested named list. The names of the list correspond to the model names. Each list element should be a list of the annotation databases tested (addAnnotations). The names of the list correspond to the annotation databases. Each list element should be another list of tests (addTests). The names correspond to the tests performed. Each of these elements should be a data frame with enrichment results. Each table must contain the following columns: "termID", "description", "nominal" (the nominal statistics), and "adjusted" (the statistics after adjusting for multiple testing). Any additional columns are ignored.

metaFeatures

The metadata variables that describe the meta-features in the study. The input object is a list of data frames (one per model). The first column of each data frame is used as the featureID, so it must contain the same IDs as the corresponding features data frame (addFeatures). To share a data frame across multiple models, use the modelID "default". All columns will be coerced to character strings.
plots

Custom plotting functions for the study. The input object is a nested list. The first list corresponds to the modelID(s). The second list corresponds to the name(s) of the function(s) defined in the current R session. The third list provides metadata to describe each plot. The only required metadata element is displayName, which controls how the plot will be named in the app. You are encouraged to also specify the plotType, e.g. "singleFeature", "multiFeature". If you do not specify the plotType, the plot will be assumed to be "singleFeature". Optionally, if the plotting function requires external packages, these can be defined in the element packages. To share plots across multiple models, use the modelID "default".

barcodes

The metadata variables that describe the barcode plot. The input object is a list of lists (one per model). Each sublist must contain the element statistic, which is the column name in the results table to use to construct the barcode plot. Each sublist may additionally contain any of the following optional elements: 1) absolute - Should the statistic be converted to its absolute value (default is TRUE). 2) logFoldChange - The column name in the results table that contains the log fold change values. 3) labelStat - The x-axis label to describe the statistic. 4) labelLow - The left-side label to describe low values of the statistic. 5) labelHigh - The right-side label to describe high values of the statistic. 6) featureDisplay - The feature variable to use to label the barcode plot on hover. To share metadata across multiple models, use the modelID "default".

reports

The analysis report(s) that explain how the study results were generated. The input object is a list of character vectors (one per model). Each element should be either a URL or a path to a file on your computer. If it is a path to a file, this file will be included in the exported study package. To share a report across multiple models, use the modelID "default".

resultsLinkouts

The URL patterns that describe linkouts to external resources (see Details below). The input object is a nested named list. The names of the list correspond to the model names. Each element of the list is a named list of character vectors. The names of this nested list must correspond to the column names of the matching features table. To share linkouts across multiple models, use the modelID "default".

enrichmentsLinkouts

The URL patterns that describe linkouts to external resources (see Details below). The input object is a named list. The names of the list correspond to the annotation names. Each element of the list is a character vector of linkouts for that annotationID.

metaFeaturesLinkouts

The URL patterns that describe linkouts to external resources (see Details below). The input object is a nested named list. The names of the list correspond to the model names. Each element of the list is a named list of character vectors. The names of this nested list must correspond to the column names of the matching metaFeatures table (addMetaFeatures). To share linkouts across multiple models, use the modelID "default".

version

(Optional) Include a version number to track the updates to your study package. If you export the study to a package, the version is used as the package version.
exportStudy

maintainer (Optional) Include the name of the study package’s maintainer
maintainerEmail (Optional) Include the email of the study package’s maintainer
studyMeta (Optional) Define metadata about your study. The input is a list of key:value pairs. See below for more details.

Details

You can add metadata to describe your study by passing a named list to the argument studyMeta. The names of the list cannot contain spaces or colons, and they can’t start with # or -. The values of each list should be a single value. Also, your metadata fields cannot use any of the reserved fields for R’s DESCRIPTION file.

Value

Returns a new OmicNavigator study object, which is a named nested list with class onStudy

See Also

addSamples, addFeatures, addModels, addAssays, addTests, addAnnotations, addResults, addEnrichments, addMetaFeatures, addPlots, addBarcodes, addReports, addResultslinkouts, addEnrichmentsLinkouts, addMetaFeaturesLinkouts, exportStudy, installStudy

Examples

study <- createStudy(name = "ABC",
                      description = "An analysis of ABC")

# Define a version and study metadata
study <- createStudy(name = "ABC",
                      description = "An analysis of ABC",
                      version = "0.1.0",
                      maintainer = "My Name",
                      maintainerEmail = "me@email.com",
                      studyMeta = list(department = "immunology",
                                        organism = "Mus musculus"))

---

**exportStudy**  
**Export a study**

Description

Export a study
Usage

```r
exportStudy(
  study,
  type = c("tarball", "package"),
  path = NULL,
  requireValid = TRUE
)
```

Arguments

- `study`: An OmicNavigator study
- `type`: Export study as a package tarball ("tarball") or as a package directory ("package")
- `path`: Optional file path to save the object
- `requireValid`: Require that study is valid before exporting

Value

Invisibly returns the name of the tarball file ("tarball") or the path to the package directory ("package")

See Also

`validateStudy`

---

getAnnotations  

Get annotations from a study

Description

Get annotations from a study

Usage

```r
getAnnotations(study, annotationID = NULL, quiet = FALSE, libraries = NULL)
```

Arguments

- `study`: An OmicNavigator study. Either an object of class `onStudy`, or the name of an installed study package.
- `annotationID`: Filter by annotation
- `quiet`: Suppress messages (default: `FALSE`)
- `libraries`: The directories to search for installed study packages. If left as `NULL` (the default), then `installed.packages` will use the result of `.libPaths`. 
getAssays

Get assays from a study

Usage

getAssays(study, modelID = NULL, quiet = FALSE, libraries = NULL)

Arguments

study An OmicNavigator study. Either an object of class onStudy, or the name of an installed study package.
modelID Filter by modelID
quiet Suppress messages (default: FALSE)
libraries The directories to search for installed study packages. If left as NULL (the default), then installed.packages will use the result of .libPaths.

Value

The object returned depends on the data available and any filters (e.g. the argument modelID):
If no filters are specified, then the object returned is a nested list, similar to the original input object.
If one or more filters are applied, then only a subset of the original nested list is returned. Technically, each filter applied is used to subset the original nested list using [[].
If no data is available, an empty list is returned (list()).
getBarcodeData

Get data for barcode and violin plots

Description

Get data for barcode and violin plots

Usage

getBarcodeData(study, modelID, testID, annotationID, termID)

Arguments

study
An OmicNavigator study. Either an object of class onStudy, or the name of an installed study package.

modelID
Filter by modelID

testID
Filter by testID

annotationID
Filter by annotation

termID
Filter by termID

Value

A list with the following components:

data
Data frame with the differential statistics to plot

highest
(numeric) The largest differential statistic, rounded up to the next integer

labelStat
(character) The x-axis label to describe the differential statistic

labelLow
(character) The vertical axis label on the left to describe smaller values (default is "Low")

labelHigh
(character) The vertical axis label on the right to describe larger values (default is "High")

See Also

addBarcodes, getBarcodes
getBarcodes

Get barcodes from a study

**Usage**

getBarcodes(study, modelID = NULL, quiet = FALSE, libraries = NULL)

**Arguments**

- **study**: An OmicNavigator study. Either an object of class `onStudy`, or the name of an installed study package.
- **modelID**: Filter by modelID
- **quiet**: Suppress messages (default: `FALSE`)
- **libraries**: The directories to search for installed study packages. If left as `NULL` (the default), then `installed.packages` will use the result of `.libPaths`.

**Value**

The object returned depends on the data available and any filters (e.g. the argument `modelID`):
- If no filters are specified, then the object returned is a nested list, similar to the original input object.
- If one or more filters are applied, then only a subset of the original nested list is returned. Technically, each filter applied is used to subset the original nested list using `[[`.
- If no data is available, an empty list is returned (`list()`).

getEnrichments

Get enrichments from a study

**Usage**

getEnrichments(study,
modelID = NULL,
annotationID = NULL,
testID = NULL,
quiet = FALSE,
libraries = NULL)

**Description**

Get enrichments from a study
getEnrichmentsIntersection

**Arguments**

- **study**: An OmicNavigator study. Either an object of class `onStudy`, or the name of an installed study package.
- **modelID**: Filter by modelID
- **annotationID**: Filter by annotation
- **testID**: Filter by testID
- **quiet**: Suppress messages (default: FALSE)
- **libraries**: The directories to search for installed study packages. If left as `NULL` (the default), then `installed.packages` will use the result of `libPaths`.

**Value**

The object returned depends on the data available and any filters (e.g. the argument `modelID`):

If no filters are specified, then the object returned is a nested list, similar to the original input object.

If one or more filters are applied, then only a subset of the original nested list is returned. Technically, each filter applied is used to subset the original nested list using `[[`.

If no data is available, an empty list is returned (`list()`).

---

**getEnrichmentsIntersection**

getEnrichmentsIntersection

**Description**

getEnrichmentsIntersection

**Usage**

```r
getEnrichmentsIntersection(
    study,
    modelID,
    annotationID,
    mustTests,
    notTests,
    sigValue,
    operator,
    type
)
```
getEnrichmentsLinkouts

Arguments

study       An OmicNavigator study. Either an object of class onStudy, or the name of an installed study package.
modelID     Filter by modelID
annotationID Filter by annotation
mustTests   The tests whose significant values must be included. (The intersection)
notTests    The tests whose significant values will be removed. (The difference)
sigValue    The significance levels for each column.
operator    The operators for each column.
type        Type of p-value ("nominal" or "adjusted")

Value

Returns a data frame with the enrichments, similar to getEnrichmentsTable. Only rows that pass all the filters are included.

See Also

getEnrichmentsTable

Description

Get enrichments table linkouts from a study

Usage

getEnrichmentsLinkouts(
  study,
  annotationID = NULL,
  quiet = FALSE,
  libraries = NULL
)

Arguments

study       An OmicNavigator study. Either an object of class onStudy, or the name of an installed study package.
annotationID Filter by annotation
quiet        Suppress messages (default: FALSE)
libraries    The directories to search for installed study packages. If left as NULL (the default), then installed.packages will use the result of .libPaths.
The object returned depends on the data available and any filters (e.g. the argument modelID):

If no filters are specified, then the object returned is a nested list, similar to the original input object.

If one or more filters are applied, then only a subset of the original nested list is returned. Techni-
cally, each filter applied is used to subset the original nested list using \[\texttt{[]}\].

If no data is available, an empty list is returned (\texttt{list()}).

---

**getEnrichmentsNetwork**  
*Get enrichments network from a study*

**Description**

Get enrichments network from a study

**Usage**

getEnrichmentsNetwork(study, modelID, annotationID, libraries = NULL)

**Arguments**

- **study**  
  An OmicNavigator study. Either an object of class \texttt{onStudy}, or the name of an 
  installed study package.

- **modelID**  
  Filter by modelID

- **annotationID**  
  Filter by annotation

- **libraries**  
  The directories to search for installed study packages. If left as NULL (the de-
  fault), then \texttt{installed.packages} will use the result of \texttt{.libPaths}.

**Value**

Returns a list with the following components:

- **tests**  
  (character) Vector of testIDs

- **nodes**  
  (data frame) The description of each annotation term (i.e. node). The nominal 
  and adjusted p-values are in list-columns.

- **links**  
  (list) The statistics for each pairwise overlap between the annotation terms (i.e. 
  nodes)
getEnrichmentsTable

Get enrichments table from a study

Description

Get enrichments table from a study

Usage

getEnrichmentsTable(
  study,
  modelID,
  annotationID,
  type = "nominal",
  libraries = NULL
)

Arguments

study An OmicNavigator study. Either an object of class onStudy, or the name of an installed study package.
modelID Filter by modelID
annotationID Filter by annotation
type Type of p-value ("nominal" or "adjusted")
libraries The directories to search for installed study packages. If left as NULL (the default), then installed.packages will use the result of .libPaths.

Value

A data frame of enrichments with the following columns:

termID The unique ID for the annotation term
description The description of the annotation term
... One column for each of the enrichments
### getEnrichmentsUpset

#### Description

getEnrichmentsUpset

#### Usage

```r
getEnrichmentsUpset(
  study,
  modelID,
  annotationID,
  sigValue,
  operator,
  type,
  tests = NULL
)
```

#### Arguments

- **study**: An OmicNavigator study. Either an object of class `onStudy`, or the name of an installed study package.
- **modelID**: Filter by modelID
- **annotationID**: Filter by annotation
- **sigValue**: The significance levels for each column.
- **operator**: The operators for each column.
- **type**: Type of p-value ("nominal" or "adjusted")
- **tests**: Restrict UpSet plot to these tests

#### Value

No return value. This function is called for the side effect of creating an UpSet plot.

### getFavicons

**Get favicon URLs for table linkouts**

#### Description

To enhance the display of the linkouts in the app’s tables, it can fetch the favicon URL for each website.
getFeatures

Get features from a study

Usage

getFeatures(study, modelID = NULL, quiet = FALSE, libraries = NULL)

Arguments

study An OmicNavigator study. Either an object of class onStudy, or the name of an
installed study package.

modelID Filter by modelID

quiet Suppress messages (default: FALSE)

libraries The directories to search for installed study packages. If left as NULL (the
default), then installed.packages will use the result of .libPaths.

Value

A data frame (if modelID is specified) or a list of data frames. All the columns will be character
strings, even if the values appear numeric.
getInstalledStudies

Description
Get installed OmicNavigator studies

Usage
getInstalledStudies(libraries = NULL)

Arguments
libraries Character vector of library directories to search for study packages. If NULL, uses .libPaths.

Value
Returns a character vector of the installed OmicNavigator study packages

Examples
getInstalledStudies()

getLinkFeatures

Description
Get the shared features in a network link

Usage
getLinkFeatures(study, annotationID, termID1, termID2)

Arguments
study An OmicNavigator study. Only accepts name of installed study package.
annotationID Filter by annotation
termID1, termID2 Linked terms to find overlapping features

Value
Returns a character vector with the features included in both termIDs (i.e. the intersection)
getMetaFeatures

Get metaFeatures from a study

Usage

getMetaFeatures(study, modelID = NULL, quiet = FALSE, libraries = NULL)

Arguments

study
An OmicNavigator study. Either an object of class onStudy, or the name of an
installed study package.

modelID
Filter by modelID

quiet
Suppress messages (default: FALSE)

libraries
The directories to search for installed study packages. If left as NULL (the
default), then installed.packages will use the result of .libPaths.

Value

The object returned depends on the data available and any filters (e.g. the argument modelID):
If no filters are specified, then the object returned is a nested list, similar to the original input object.
If one or more filters are applied, then only a subset of the original nested list is returned. Techni-
cally, each filter applied is used to subset the original nested list using [[].
If no data is available, an empty list is returned (list()).

getMetaFeaturesLinkouts

Get metaFeatures table linkouts from a study

Usage

getMetaFeaturesLinkouts(study, modelID = NULL, quiet = FALSE, libraries = NULL)
Arguments

study An OmicNavigator study. Either an object of class onStudy, or the name of an installed study package.

modelID Filter by modelID

quiet Suppress messages (default: FALSE)

libraries The directories to search for installed study packages. If left as NULL (the default), then installed.packages will use the result of .libPaths.

Value

The object returned depends on the data available and any filters (e.g. the argument modelID):

If no filters are specified, then the object returned is a nested list, similar to the original input object.

If one or more filters are applied, then only a subset of the original nested list is returned. Technically, each filter applied is used to subset the original nested list using [[].

If no data is available, an empty list is returned (list()).

getMetaFeaturesTable  

Get metaFeatures for a given feature

Usage

getMetaFeaturesTable(study, modelID, featureID)

Arguments

study An OmicNavigator study. Either an object of class onStudy, or the name of an installed study package.

modelID Filter by modelID

featureID Filter by featureID

Value

Returns a data frame with the metaFeatures for the provided featureID. If the featureID is not found in the metaFeatures table, the data frame will have zero rows.

See Also

addMetaFeatures, getMetaFeatures
getModels

Description

Get models from a study

Usage

getModels(study, modelID = NULL, quiet = FALSE, libraries = NULL)

Arguments

study An OmicNavigator study. Either an object of class onStudy, or the name of an installed study package.
modelID Filter by modelID
quiet Suppress messages (default: FALSE)
libraries The directories to search for installed study packages. If left as NULL (the default), then installed.packages will use the result of .libPaths.

Value

The object returned depends on the data available and any filters (e.g. the argument modelID):

If no filters are specified, then the object returned is a nested list, similar to the original input object.

If one or more filters are applied, then only a subset of the original nested list is returned. Technically, each filter applied is used to subset the original nested list using [[].

If no data is available, an empty list is returned (list()).

getNodeFeatures

Get the features in a network node

Description

Get the features in a network node

Usage

getNodeFeatures(study, annotationID, termID, libraries = NULL)
getOverlaps

Arguments

study An OmicNavigator study. Only accepts name of installed study package.
annotationID Filter by annotation
termID Filter by termID
libraries The directories to search for installed study packages. If left as NULL (the
default), then \texttt{installed.packages} will use the result of \texttt{.libPaths}.

Value

Returns a character vector with the features in the termID

See Also

\texttt{getLinkFeatures}

---

getOverlaps \textit{Get overlaps from a study}

Description

Get overlaps from a study

Usage

\texttt{getOverlaps(study, annotationID = NULL, quiet = FALSE, libraries = NULL)}

Arguments

study An OmicNavigator study. Either an object of class \texttt{onStudy}, or the name of an
installed study package.
annotationID Filter by annotation
quiet Suppress messages (default: \texttt{FALSE})
libraries The directories to search for installed study packages. If left as NULL (the
default), then \texttt{installed.packages} will use the result of \texttt{.libPaths}.

Value

The object returned depends on the data available and any filters (e.g. the argument \texttt{modelID}):
If no filters are specified, then the object returned is a nested list, similar to the original input object.
If one or more filters are applied, then only a subset of the original nested list is returned. Techni-
cally, each filter applied is used to subset the original nested list using \texttt{[[].}
If no data is available, an empty list is returned (\texttt{list()}).
**getPackageVersion**  
*Get version of OmicNavigator package*

**Description**

This is a convenience function for the app. It is easier to always call the OmicNavigator package functions via OpenCPU than to call the utils package for this one endpoint.

**Usage**

```r
getPackageVersion()
```

**Value**

Returns a one-element character vector with the version of the currently installed OmicNavigator R package.

---

**getPlots**  
*Get plots from a study*

**Description**

Get plots from a study

**Usage**

```r
getPlots(study, modelID = NULL, quiet = FALSE, libraries = NULL)
```

**Arguments**

- **study**: An OmicNavigator study. Either an object of class `onStudy`, or the name of an installed study package.
- **modelID**: Filter by modelID
- **quiet**: Suppress messages (default: `FALSE`)
- **libraries**: The directories to search for installed study packages. If left as `NULL` (the default), then `installed.packages` will use the result of `.libPaths`.

**Value**

The object returned depends on the data available and any filters (e.g. the argument `modelID`):

- If no filters are specified, then the object returned is a nested list, similar to the original input object.
- If one or more filters are applied, then only a subset of the original nested list is returned. Technically, each filter applied is used to subset the original nested list using `[]`.
- If no data is available, an empty list is returned (`list()`)
getPlottingData

*Description*

This function creates the input data that `plotStudy` passes to custom plotting functions added with `addPlots`. You can use it directly when you are interactively creating your custom plotting functions.

*Usage*

```r
getPlottingData(study, modelID, featureID, libraries = NULL)
```

*Arguments*

- **study**: An OmicNavigator study. Either an object of class `onStudy`, or the name of an installed study package.
- **modelID**: Filter by modelID
- **featureID**: Filter by featureID
- **libraries**: The directories to search for installed study packages. If left as `NULL` (the default), then `installed.packages` will use the result of `.libPaths`.

*Value*

Returns a list of 3 data frames:

- **assays**: A data frame that contains the assay measurements, filtered to only include the row(s) corresponding to the input featureID(s) (see `getAssays`). If multiple featureIDs are requested, the rows are reordered to match the order of this input. The column order is unchanged.
- **samples**: A data frame that contains the sample metadata for the given modelID (see `getSamples`). The rows are reordered to match the columns of the assays data frame.
- **features**: A data frame that contains the feature metadata, filtered to only include the row(s) corresponding to the input featureID(s) (see `getFeatures`). If multiple featureIDs are requested, the rows are reordered to match the order of this input (and thus match the order of the assays data frame).

*See Also*

`addPlots`, `plotStudy`
getReportLink  

Get link to report

Description

Get link to report

Usage

getReportLink(study, modelID)

Arguments

study  
An OmicNavigator study. Either an object of class onStudy, or the name of an installed study package.

modelID  
Filter by modelID

Value

Returns a one-element character vector with either a path to a report file or a URL to a report web page. If no report is available for the modelID, an empty character vector is returned.

getReports  
Get reports from a study

Description

Get reports from a study

Usage

getReports(study, modelID = NULL, quiet = FALSE, libraries = NULL)

Arguments

study  
An OmicNavigator study. Either an object of class onStudy, or the name of an installed study package.

modelID  
Filter by modelID

quiet  
Suppress messages (default: FALSE)

libraries  
The directories to search for installed study packages. If left as NULL (the default), then installed.packages will use the result of .libPaths.
getResults

Value

The object returned depends on the data available and any filters (e.g. the argument modelID):
If no filters are specified, then the object returned is a nested list, similar to the original input object.
If one or more filters are applied, then only a subset of the original nested list is returned. Technically, each filter applied is used to subset the original nested list using `[]`.
If no data is available, an empty list is returned (`list()`).

---

getResults

Get results from a study

Description

Get results from a study

Usage

getResults(
  study,
  modelID = NULL,
  testID = NULL,
  quiet = FALSE,
  libraries = NULL
)

Arguments

- **study**: An OmicNavigator study. Either an object of class `onStudy`, or the name of an installed study package.
- **modelID**: Filter by modelID
- **testID**: Filter by testID
- **quiet**: Suppress messages (default: `FALSE`)
- **libraries**: The directories to search for installed study packages. If left as `NULL` (the default), then `installed.packages` will use the result of `.libPaths`.

Value

The object returned depends on the data available and any filters (e.g. the argument modelID):
If no filters are specified, then the object returned is a nested list, similar to the original input object.
If one or more filters are applied, then only a subset of the original nested list is returned. Technically, each filter applied is used to subset the original nested list using `[]`.
If no data is available, an empty list is returned (`list()`).
getResultsIntersection

**Description**

getResultsIntersection

**Usage**

```r
getResultsIntersection(  
  study,  
  modelID,  
  anchor,  
  mustTests,  
  notTests,  
  sigValue,  
  operator,  
  column  
)
```

**Arguments**

- **study**: An OmicNavigator study. Either an object of class `onStudy`, or the name of an installed study package.
- **modelID**: Filter by modelID
- **anchor**: The primary test to filter from.
- **mustTests**: The tests whose significant values must be included. (The intersection)
- **notTests**: The tests whose significant values will be removed. (The difference)
- **sigValue**: The significance levels for each column.
- **operator**: The operators for each column.
- **column**: The columns to be thresholded.

**Value**

Returns a data frame with the results, similar to `getResultsTable`. Only rows that pass all the filters are included. The new column `Set_Membership` is a comma-separated field that includes the testIDs in which the featureID passed the filters.

**See Also**

`getResultsTable`
getResultsLinkouts  

*Get results table linkouts from a study*

**Description**

Get results table linkouts from a study

**Usage**

```r
getResultsLinkouts(study, modelID = NULL, quiet = FALSE, libraries = NULL)
```

**Arguments**

- **study**: An OmicNavigator study. Either an object of class `onStudy`, or the name of an installed study package.
- **modelID**: Filter by modelID
- **quiet**: Suppress messages (default: `FALSE`)
- **libraries**: The directories to search for installed study packages. If left as `NULL` (the default), then `installed.packages` will use the result of `.libPaths`.

**Value**

The object returned depends on the data available and any filters (e.g. the argument `modelID`):

- If no filters are specified, then the object returned is a nested list, similar to the original input object.
- If one or more filters are applied, then only a subset of the original nested list is returned. Technically, each filter applied is used to subset the original nested list using `[[`.
- If no data is available, an empty list is returned (`list()`).

getResultsTable  

*Get results table from a study*

**Description**

Get results table from a study

**Usage**

```r
getResultsTable(study, modelID, testID, libraries = NULL)
```
getResultsUpset

Arguments

study An OmicNavigator study. Either an object of class onStudy, or the name of an installed study package.

modelID Filter by modelID

testID Filter by testID

libraries The directories to search for installed study packages. If left as NULL (the default), then installed.packages will use the result of .libPaths.

Value

A data frame which includes the columns from the features table followed by the columns from the results table. All the columns from the features table will be character strings, even if the values appear numeric.

getResultsUpset

getResultsUpset

Description

getResultsUpset

Usage

getResultsUpset(study, modelID, sigValue, operator, column, legacy = FALSE)

Arguments

study An OmicNavigator study. Either an object of class onStudy, or the name of an installed study package.

modelID Filter by modelID

sigValue The significance levels for each column.

operator The operators for each column.

column The columns to be thresholded.

legacy Use legacy code (for testing purposes only)

Value

Invisibly returns the output from upset
getSamples

Get samples from a study

Usage

gtSamples(study, modelID = NULL, quiet = FALSE, libraries = NULL)

Arguments

study An OmicNavigator study. Either an object of class onStudy, or the name of an installed study package.
modelID Filter by modelID
quiet Suppress messages (default: FALSE)
libraries The directories to search for installed study packages. If left as NULL (the default), then installed.packages will use the result of .libPaths.

Value

The object returned depends on the data available and any filters (e.g. the argument modelID):
If no filters are specified, then the object returned is a nested list, similar to the original input object.
If one or more filters are applied, then only a subset of the original nested list is returned. Technically, each filter applied is used to subset the original nested list using [[]].
If no data is available, an empty list is returned (list()).

getTests

Get tests from a study

Usage

gtTests(study, modelID = NULL, testID = NULL, quiet = FALSE, libraries = NULL)
getUpsetCols

Arguments

study An OmicNavigator study. Either an object of class onStudy, or the name of an installed study package.

modelID Filter by modelID

testID Filter by testID

quiet Suppress messages (default: FALSE)

libraries The directories to search for installed study packages. If left as NULL (the default), then installed.packages will use the result of .libPaths.

Value

The object returned depends on the data available and any filters (e.g. the argument modelID):
If no filters are specified, then the object returned is a nested list, similar to the original input object.
If one or more filters are applied, then only a subset of the original nested list is returned. Technically, each filter applied is used to subset the original nested list using [[].
If no data is available, an empty list is returned (list()).

getUpsetCols

Description

Determine the common columns across all tests of a model that are available for filtering with UpSet.

Usage

getUpsetCols(study, modelID)

Arguments

study An OmicNavigator study. Either an object of class onStudy, or the name of an installed study package.

modelID Filter by modelID

Value

Returns a character vector with the names of the common columns
group

Description

A subset of the object group from Bioconductor workflow RNAseq123.

Usage

group

Format

A factor with 3 levels:

- **Basal**: Basal cells
- **LP**: Luminal progenitor cells
- **ML**: Mature luminal cells

Source


References


Examples

table(group)
str(group)
importStudy  

*Import a study package*

**Description**

Create an onStudy object by importing an installed study package

**Usage**

```r
importStudy(study, libraries = NULL)
```

**Arguments**

- **study**: Named of an installed OmicNavigator study
- **libraries**: The directories to search for installed study packages. If left as NULL (the default), then `installed.packages` will use the result of `.libPaths`.

**Value**

Returns the onStudy object imported from the OmicNavigator study package

---

installApp  

*Install the OmicNavigator web app*

**Description**

In order to run the OmicNavigator web app on your local machine, the app must be installed in the www/ subdirectory of the R package. If you installed the release tarball from the GitHub Releases page, then you already have the app installed. If you installed directly from GitHub with `install_github`, or if you want to use a different version of the app, you can manually download and install the app.

**Usage**

```r
installApp(version = NULL, overwrite = FALSE, lib.loc = NULL, ...)
```

**Arguments**

- **version**: Version of the web app to install, e.g. "1.0.0"
- **overwrite**: Should an existing installation of the app be overwritten?
- **lib.loc**: a character vector with path names of R libraries. See ‘Details’ for the meaning of the default value of NULL.
- **...**: Passed to `download.file`. If the download fails, you may need to adjust the download settings for your operating system. For example, to download with `wget`, pass the argument `method = "wget"`. 
installStudy  

Description
Install a study as an R package

Usage
installStudy(study, library = .libPaths()[1])

Arguments
study  
An OmicNavigator study to install (class onStudy)

library  
Directory to install package. Defaults to first directory returned by .libPaths.

Value
Invisibly returns the original onStudy object that was passed to the argument study

lane  

Description
A subset of the object lane from Bioconductor workflow RNAseq123.

Usage
lane

Format
A factor with 3 levels:

- L004  Sample sequenced on lane 4
- L006  Sample sequenced on lane 6
- L008  Sample sequenced on lane 8
**lcpm**

**Source**


**References**


**Examples**

```r
table(lane)
str(lane)
```

<table>
<thead>
<tr>
<th>lcpm</th>
<th>lcpm from Bioconductor workflow RNAseq123</th>
</tr>
</thead>
</table>

**Description**

A subset of the object lcpm from Bioconductor workflow RNAseq123.

**Usage**

```
lcpm
```

**Format**

A matrix with 24 rows and 9 columns

**Source**


**References**


Examples

head(lcpm)
str(lcpm)

listStudies

List available studies and their metadata

Description

List available studies and their metadata

Usage

listStudies(libraries = NULL)

Arguments

libraries The directories to search for installed study packages. If left as NULL (the default), then \texttt{installed.packages} will use the result of \texttt{.libPaths}.

Value

Returns a nested list with one element per installed OmicNavigator study package. Each study package entry has the following sublist components:

- **name**: (character) Name of the study
- **package**: (list) The fields from DESCRIPTION
- **results**: (nested list) The testIDs available for each modelID
- **enrichments**: (nested list) The annotationIDs available for each modelID
- **plots**: (nested list) The plotIDs available for each modelID

Mm.c2

Mm.c2 from Bioconductor workflow RNAseq123

Description

A subset of the object \texttt{Mm.c2} from Bioconductor workflow RNAseq123.

Usage

Mm.c2
Format

A list of 4 character vectors

Source


References


Examples

```r
Mm.c2[[1]]
str(Mm.c2)
```

Description

Package options to control package-wide behavior are described below.

Details

The default prefix for OmicNavigator study packages is "ONstudy". If you would prefer to use a different prefix, you can change the package option `OmicNavigator.prefix`. For example, to use the prefix "OmicNavigatorStudy", you could add the following line to your `.Rprofile` file:

```r
options(OmicNavigator.prefix = "OmicNavigatorStudy")
```
plotStudy  

*Plot a feature using a custom plotting function*

**Description**

Plot a feature using a custom plotting function

**Usage**

```r
plotStudy(study, modelID, featureID, plotID, libraries = NULL)
```

**Arguments**

- **study**: An OmicNavigator study. Either an object of class `onStudy`, or the name of an installed study package.
- **modelID**: Filter by modelID
- **featureID**: Filter by featureID
- **plotID**: Filter by plotID
- **libraries**: The directories to search for installed study packages. If left as NULL (the default), then `installed.packages` will use the result of `.libPaths`.

**Value**

This function is called for the side effect of creating a plot. However, it also invisible returns the original `onStudy` object passed to `study`.

**See Also**

- `addPlots`, `getPlottingData`

---

removeStudy  

*Remove an installed study R package*

**Description**

Remove an installed study R package

**Usage**

```r
removeStudy(study, library = .libPaths()[1])
```
Arguments

study The name of the study or an onStudy object. Do not include the prefix of the installed package, e.g. ONstudy.

library Directory where the study package is installed. Defaults to first directory returned by .libPaths.

Value

Invisibly returns the path of the removed study package

---

**samplenames**

*samplenames from Bioconductor workflow RNAseq123*

Description

A subset of the object samplenames from Bioconductor workflow RNAseq123.

Usage

samplenames

Format

A character vector containing the unique sample identifiers

Source


References


Examples

head(samplenames)
str(samplenames)
**startApp**  
*Start app on local machine*

**Description**

After you have installed at least one OmicNavigator study package with `installStudy`, you can explore the results in the app. The function `startApp` starts a local instance of the app running on your current machine. It will automatically open the app in your default browser. For the best experience, use Google Chrome. From the dropdown menu, you will be able to select from any of the studies you have installed on your machine. When you are finished, you can stop the web server by returning to the R console and pressing the Esc key (Windows) or Ctrl-C (Linux, macOS).

**Usage**

```r
startApp(...)
```

**Arguments**

- `...` extra parameters passed to `ocpu_start_server`  

**Details**

Note that the app can’t be run from within RStudio Server.

The app requires some additional R packages to run. If you receive an error about a missing package, please install it with `install.packages`. To ensure you have all the extra packages installed, you can run the command below:

```r
install.packages(c("faviconPackage", "opencpu", "UpSetR"))
```

**Value**

No return value. This function is only called for the side effect of running a local instance of the app.

---

**summary.onStudy**  
*Summarize elements of OmicNavigator study*

**Description**

Displays a tree-like summary of the elements that have been added to an OmicNavigator study.

**Usage**

```r
## S3 method for class 'onStudy'
summary(object, elements = NULL, ...)
```
validateStudy

Arguments

- object: OmicNavigator study object (class onStudy)
- elements: Subset the output to only include specific elements of the study, e.g. c("results", "enrichments")
- ...: Currently unused

Value

Invisibly returns the original onStudy object

validateStudy(study)

Description

Validate a study

Usage

validateStudy(study)

Arguments

- study: An OmicNavigator study object

Value

For a valid study object, the logical value TRUE is invisibly returned. For an invalid study object, there is no return value because an error is thrown.
Index

* datasets
  - basal.vs.lp, 18
  - basal.vs.ml, 19
  - cam.BasalvsLP, 20
  - cam.BasalvsML, 21
  - group, 50
  - lane, 52
  - lcpm, 53
  - Mm.c2, 54
  - samplenames, 57

.libPaths, 26, 27, 29–33, 34, 35, 37–44, 46–49, 51, 52, 54, 56, 57
[[, 27, 29, 30, 32, 37–41, 44, 46, 48, 49

addAnnotations, 3, 6, 7, 23, 25
addAssays, 4, 25
addBarcodes, 5, 25, 28
addEnrichments, 6, 7, 25
addEnrichmentPatterns, 6, 25
addFeatures, 4, 8, 9, 15, 23, 25
addMetaFeatures, 8, 9, 10, 24, 25, 38
addMetaFeaturesLinkouts, 9, 25
addModels, 10, 25
addOverlaps, 11
addPlots, 12, 25, 42, 56
addReports, 13, 25
addResults, 14, 25
addResultsLinkouts, 14, 25
addSamples, 4, 16, 23, 25
addTests, 6, 16, 23, 25

basal.vs.lp, 18
basal.vs.ml, 19

cam.BasalvsLP, 20
cam.BasalvsML, 21
createStudy, 4–9, 11–17, 22

download.file, 51
exportStudy, 25, 25

getAnnotations, 26
getAssays, 27, 29
getBarcodeData, 28
getBarcodes, 28, 29
getEnrichments, 29
getEnrichmentsIntersection, 30
getEnrichmentsLinkouts, 31, 35
getEnrichmentsNetwork, 32
getEnrichmentsTable, 31, 33
getEnrichmentsUpset, 34
getFavicons, 34
getFeatures, 35, 42
getInstalledStudies, 36
getLinkFeatures, 36, 40
getMetaFeatures, 37, 38
getMetaFeaturesLinkouts, 37
getMetaFeaturesTable, 38
getModels, 39
getNodeFeatures, 37, 39
getOverlaps, 40
getPackageVersion, 41
getPlots, 41
getPlottingData, 13, 42, 56
getReportLink, 43
getReports, 43
getResults, 44
getResultsIntersection, 45
getResultsLinkouts, 35, 46
getResultsTable, 45, 46
getResultsUpset, 47
getSamples, 42, 48
getTests, 48
getUpsetCols, 49

importStudy, 51
install.packages, 58
installApp, 51
installed.packages, 26, 27, 29–33, 34, 37–44, 46–49, 51, 54, 56
installStudy, 25, 52, 58
lane, 52
lcpm, 53
listStudies, 54
Mm.c2, 54
ocpu_start_server, 58
OmicNavigator, 55
plotStudy, 13, 42, 56
removeStudy, 56
samplenames, 57
startApp, 58
summary.onStudy, 58
upset, 47
validateStudy, 26, 59