# Package ‘OmicNavigator’

December 17, 2025

**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** 1.19.0

**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, plotly, tinytest (>= 1.2.3), ttdo (>= 0.0.6), UpSetR

**RoxygenNote** 7.3.3

**NeedsCompilation** no

**Author** Terrence Ernst [aut] (Web application),
John Blischak [aut] (ORCID: <`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, cre],
Marco Curado [aut] (Improved plotting capabilities),
Joe LoGrasso [aut] (Support for plotly),
Elyse Geoffroy [ctb],
AbbVie Inc. [cph, fnd]

**Maintainer** Brett Engelmann <`brett.engelmann@abbvie.com`>

**Repository** CRAN

**Date/Publication** 2025-12-17 07:50:08 UTC

## Contents

---

`OmicNavigator-package` *OmicNavigator*

---

## 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.

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

## Author(s)

**Maintainer**: Brett Engelmann `<brett.engelmann@abbvie.com>`

Authors:

- Terrence Ernst (Web application)
- John Blischak `<jdblischak@gmail.com>` (ORCID)
- Paul Nordlund (Web application)
- Justin Moore (UpSet-related functions and web application)
- Joe Dalen (Barcode functionality and web application)
- Akshay Bhamidipati (Web application)
- Marco Curado (Improved plotting capabilities)
- Joe LoGrasso (Support for plotly)

Other contributors:

- Elyse Geoffroy [contributor]
- AbbVie Inc. [copyright holder, funder]

## See Also

Useful links:

- `https://github.com/abbvie-external/OmicNavigator`
- Report bugs at `https://github.com/abbvie-external/OmicNavigator/issues`

## addAnnotations *Add annotations*

### 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 a list 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. |
| `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

### See Also

`getAnnotations`

## addAssays *Add assays*

### 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. |

### Value

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

### See Also

`getAssays`

---

## `addBarcodes` *Add barcode plot metadata*

---

### 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

```
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:<br>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

## See Also

`getBarcodes`

---

`addEnrichments` *Add enrichment results*

---

## Description

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

## See Also

`getEnrichments`

---

`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)
```

## 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:

```
go = "https://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:

```
reactome = "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

`getEnrichmentsLinkouts`, `addAnnotations`, `addEnrichments`

### Examples

```
study <- createStudy("example")
enrichmentsLinkouts <- list(
  gobp = c("https://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*

### Description

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

## See Also

`getFeatures`

---

`addMapping` *Add mapping object*

---

## Description

Add mapping object

## Usage

```
addMapping(study, mapping, reset = FALSE)
```

## Arguments

| | |
|---|---|
| `study` | An OmicNavigator study created with `createStudy` |
| `mapping` | Feature IDs from models. The input object is a list of named data frames. For each data frame, column names indicate model names while rows indicate featureIDs per model. Features with same index position across columns are treated as mapped across models. For each model, feature IDs must match feature IDs available in the results object of the respective model. 1:N relationships are allowed.<br>Mapping list elements are required to be named as 'default' or after a model name as provided in addModels(). If a single data frame is provided, this list element is recommended to be named 'default'. For multiple list elements, each with its own data frame, list elements should be named after model name(s) (a single element may still be named 'default'). In that case, when navigating in ON front-end (FE), mapping element related to the selected model in the FE will be used in multimodel plots. If a selected model in FE does not have a corresponding mapping list element, it may still use the mapping list element called 'default' if this is available.<br>E.g., if in a study there are models "transcriptomics" and "proteomics" and the user wants to create a plot based on data from both, a mapping list should be provided with addMapping(). In this case, the mapping list element may be named 'default'. This should contain a data frame with column names 'transcriptomics' and 'proteomics', where feature IDs that map across models are found in the same row. |
| `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

### See Also

`getMapping`, `getPlottingData`, `plotStudy`

---

## `addMetaAssays` *Add metaAssays*

---

### Description

Experimental. Add metaAssay measurements that map to the metaFeatureIDs in the metaFeatures table.

### Usage

```
addMetaAssays(study, metaAssays, reset = FALSE)
```

### Arguments

| | |
|---|---|
| `study` | An OmicNavigator study created with `createStudy` |
| `metaAssays` | The metaAssays from the study. The input object is a list of data frames (one per model). The row names should correspond to the metaFeatureIDs (second column of data frame added via `addMetaFeatures`). 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. |

### See Also

`getMetaAssays`, `addAssays`, `addMetaFeatures`

---

## `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`). The second column of each data frame is used as the metaFeatureID, and thus should match the row names of any metaAssays added via `addMetaAssays`. 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

## See Also

`getMetaFeatures`

---

`addMetaFeaturesLinkouts` *Add linkouts to external resources in the metaFeatures table*

---

## 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:

```
entrez = "https://www.ncbi.nlm.nih.gov/gene/"
```

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

`getMetaFeaturesLinkouts`, `addMetaFeatures`

## Examples

```
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="),
    entrez = "https://www.ncbi.nlm.nih.gov/gene/"
  )
)
study <- addMetaFeaturesLinkouts(study, metaFeaturesLinkouts)
```

## addModels *Add models*

### Description

Add models

### Usage

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

### 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

### See Also

`getModels`

### Examples

```
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"
  )
)
```

## addObjects *Add objects*

### Description

Experimental. Add arbitrary R objects to a study. These will be exported via `saveRDS` and imported via `readRDS`. This allows preserving the exact structure of complex R objects.

### Usage

```
addObjects(study, objects, reset = FALSE)
```

### Arguments

| | |
|---|---|
| `study` | An OmicNavigator study created with `createStudy` |
| `objects` | Any arbitrary R objects from the study. The input object is a list of objects (one per model). To share an object 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

The main purpose of adding a custom object to your study package is to use it in custom plots in the app. If available, they will be returned by `getPlottingData`. If the custom package requires additional R packages to be available to use, make sure to list these packages in the field packages when adding the custom plotting function via `addPlots`.

### See Also

`getObjects`, `saveRDS`, `readRDS`

## 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

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

### Arguments

| | |
|---|---|
| `study` | An OmicNavigator study created with `createStudy` |
| `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

### See Also

`getOverlaps`

## addPlots *Add custom plotting functions*

### Description

`addPlots()` adds custom plotting functions and plot metadata to an OmicNavigator study.

### Usage

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

## Arguments

| | |
|---|---|
| `study` | An OmicNavigator study created with `createStudy` |
| `plots` | A nested list containing custom plotting functions and plot metadata. The input object is a 3-level nested list. The first, or top-level list element name(s) must match the study `modelID`(s). The second, or mid-level list element name(s) must match the names of the plotting function(s) defined in the current R session (see Details below for function construction requirements). The third, or bottom-level list provides metadata to categorize, display, and support each plot. The accepted fields are `displayName`, `description`, `plotType`, `models`, and `packages`. `displayName` sets the plot name in the app and the `description` field will display as a tool tip when hovering over plotting dropdown menus. The `plotType` field is a character vector that categorizes the plot by 1) the number of features it supports (“`singleFeature`” or “`multiFeature`”), 2) the number of test results used by the plotting function (“`singleTest`”, “`multiTest`”), 3) if data from one or more models is used (add “`multiModel`” to specify that data from two or more models are used in the plot; otherwise the plot is assumed to reference only data within the model specified by the top-level list element name), and 4) if the plot is interactive (add “`plotly`” to specify interactive plots built using the plotly package; otherwise the plot is assumed to be static). e.g., `plotType = c("multiFeature", "multiTest", "plotly")`. If you do not specify the `plotType`, the plot will be designated as `plotType = c("singleFeature", "singleTest")`. The `models` field is an optional character vector that specifies the models that should be used by the app when invoking your custom plotting function. This field is set to ‘all’ by default and is only used when `plotType` includes “`multiModel`”. If this field is not included the app will assume all models in the study should be used with your plotting function. If the plotting function requires additional packages beyond those attached by default to a fresh R session, these must be defined in the element `packages`. To share a plotting functions across multiple models, use the modelID "default". Alternatively, to share a plot across a specific subset of models, you can explicitly add the same plotting function to each model (option available as of OmicNavigator 1.16.0). |
| `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

Custom plotting functions must be constructed to accept as the first argument the value returned from `getPlottingData()`. Custom plotting functions can have additional arguments, but these must be provided with default values. The end-user should call `getPlottingData()` when testing their custom plotting function. The end-user should consider the nature of the plot, i.e. the `plotType` and (rarely) `models` values (see `getPlottingData()`). For example, a custom plotting function meant to produce a `multiTest` plot should accept the output of a `getPlottingData()` call with multiple `testID`s assigned to the `testID` argument. See the details section of `plotStudy()` for a description of how `plotType` dictates the way a custom plotting function is invoked by the app.

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.

Note that the plotting functions are written to the R package when the study is exported via `exportStudy` or installed via `installStudy`, not when `addPlots` is invoked. In other words, if you add a custom plotting function to your study object via `addPlots`, but then subsequently update the function in the global environment prior to installing the study, this latest version will be saved in the R package and executed when run in the app.

## Value

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

## See Also

`getPlots`, `getPlottingData`, `plotStudy`

---

`addReports` *Add reports*

---

## Description

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

## Usage

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

## Arguments

| | |
|---|---|
| `study` | An OmicNavigator study created with `createStudy` |
| `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". |
| `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

## See Also

`getReports`

---

`addResults` *Add inference results*

---

### Description

Add inference results

### Usage

```
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

### See Also

`getResults`

## addResultsLinkouts *Add linkouts to external resources in the results table*

### 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

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

### Arguments

| | |
|---|---|
| `study` | An OmicNavigator study created with `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: `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 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:

```
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:

```
entrez = "https://www.ncbi.nlm.nih.gov/gene/"
```

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

getResultsLinkouts, addFeatures

### Examples

```
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="),
    entrez = "https://www.ncbi.nlm.nih.gov/gene/"
  )
)
study <- addResultsLinkouts(study, resultsLinkouts)
```

---

## `addSamples` *Add sample metadata*

---

### Description

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

### See Also

getSamples

## addTests *Add tests*

### Description

Add tests

### Usage

```
addTests(study, tests, reset = FALSE)
```

### 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. Furthermore, any fields that match the column names in the results table (added via addFeatures or addResults) will be used to derive tooltips for those columns. |
| 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

### See Also

getTests

### Examples

```
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"
    )
  )
)
```

## `basal.vs.lp` *basal.vs.lp from Bioconductor workflow RNAseq123*

### 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

https://bioconductor.org/packages/release/workflows/vignettes/RNAseq123/inst/doc/limmaWorkflow.html

### References

Law CW, Alhamdoosh M, Su S, Dong X, Tian L, Smyth GK, Ritchie ME. RNA-seq analysis is easy as 1-2-3 with limma, Glimma and edgeR [version 3; peer review: 3 approved]. F1000Research 2018, 5:1408 doi:10.12688/f1000research.9005.3

Sheridan, J.M., Ritchie, M.E., Best, S.A. et al. A pooled shRNA screen for regulators of primary mammary stem and progenitor cells identifies roles for *Asap1* and *Prox1*. BMC Cancer 2015, 15:221 doi:10.1186/s128850151187z

### Examples

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

---

## basal.vs.ml — *basal.vs.ml from Bioconductor workflow RNAseq123*

---

### 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

`https://bioconductor.org/packages/release/workflows/vignettes/RNAseq123/inst/doc/limmaWorkflow.html`

### References

Law CW, Alhamdoosh M, Su S, Dong X, Tian L, Smyth GK, Ritchie ME. RNA-seq analysis is easy as 1-2-3 with limma, Glimma and edgeR [version 3; peer review: 3 approved]. F1000Research 2018, 5:1408 doi:10.12688/f1000research.9005.3

Sheridan, J.M., Ritchie, M.E., Best, S.A. et al. A pooled shRNA screen for regulators of primary mammary stem and progenitor cells identifies roles for *Asap1* and *Prox1*. BMC Cancer 2015, 15:221 doi:10.1186/s128850151187z

### Examples

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

---

## `cam.BasalvsLP` *cam.BasalvsLP from Bioconductor workflow RNAseq123*

---

### 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

`https://bioconductor.org/packages/release/workflows/vignettes/RNAseq123/inst/doc/limmaWorkflow.html`

### References

Law CW, Alhamdoosh M, Su S, Dong X, Tian L, Smyth GK, Ritchie ME. RNA-seq analysis is easy as 1-2-3 with limma, Glimma and edgeR [version 3; peer review: 3 approved]. F1000Research 2018, 5:1408 doi:10.12688/f1000research.9005.3

Sheridan, J.M., Ritchie, M.E., Best, S.A. et al. A pooled shRNA screen for regulators of primary mammary stem and progenitor cells identifies roles for *Asap1* and *Prox1*. BMC Cancer 2015, 15:221 doi:10.1186/s128850151187z

## Examples

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

---

`cam.BasalvsML` *cam.BasalvsML from Bioconductor workflow RNAseq123*

---

## 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

https://bioconductor.org/packages/release/workflows/vignettes/RNAseq123/inst/doc/limmaWorkflow.html

## References

Law CW, Alhamdoosh M, Su S, Dong X, Tian L, Smyth GK, Ritchie ME. RNA-seq analysis is easy as 1-2-3 with limma, Glimma and edgeR [version 3; peer review: 3 approved]. F1000Research 2018, 5:1408 doi:10.12688/f1000research.9005.3

Sheridan, J.M., Ritchie, M.E., Best, S.A. et al. A pooled shRNA screen for regulators of primary mammary stem and progenitor cells identifies roles for *Asap1* and *Prox1*. BMC Cancer 2015, 15:221 doi:10.1186/s128850151187z

## Examples

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

`combineStudies` *Combine two or more studies*

## Description

Create a new OmicNavigator study by combining two or more existing study objects.

## Usage

```
combineStudies(...)
```

## Arguments

| | |
|---|---|
| `...` | Two or more objects of class `onStudy` |

## Details

This is a convenience function to quickly and conveniently combine studies. However, it is naive, and you will likely need to edit the new study after combining. When there are conflicting elements (e.g. different study names or different maintainers), then the value for the latter study is kept. As a concrete example, if you combined 5 studies, the name of the combined study would be the name of the 5th study.

The behavior is more complex for study elements that are nested lists of data frames (e.g. results). If the 5 studies included a results table for the same modelID/testID combination, then only the results from the 5th study would be retained. However, if they each defined a different modelID, then the results for all 5 modelIDs would be included in the combined study. Please note that you should be extra cautious in the situation where the studies have the same modelID/testID combination. Ideally they should all have the same column names. Since a data frame is technically a list, the workhorse function `modifyList` will retain any uniquely named columns from earlier studies along with the columns from the final study.

Note that as a shortcut you can also combine studies using the S3 method `c`.

If a study you would like to combine is already installed, you can convert it to a study object by importing it with `importStudy`.

## Value

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

## See Also

`createStudy`, `importStudy`

## Examples

```
# Define threee study objects
studyOne <- createStudy(name = "One",
                        description = "First study",
                        studyMeta = list(metafield1 = "metavalue1"))

studyTwo <- createStudy(name = "Two",
                        description = "Second study",
                        maintainer = "The Maintainer",
                        studyMeta = list(metafield2 = "metavalue2"))

studyThree <- createStudy(name = "Three",
                          description = "Third study",
                          studyMeta = list(metafield3 = "metavalue3"))

# Combine the three studies
combineStudies(studyOne, studyTwo, studyThree)

# Equivalently, can use c()
c(studyOne, studyTwo, studyThree)
```

---

`createStudy` *Create a study*

---

## Description

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(),
  mapping = list(),
  barcodes = list(),
  reports = list(),
  resultsLinkouts = list(),
  enrichmentsLinkouts = list(),
  metaFeaturesLinkouts = list(),
  metaAssays = list(),
  objects = 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. Furthermore, any fields that match the column names in the results table (added via `addFeatures` or `addResults`) will be used to derive tooltips for those columns. |
| `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 a list 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`). The second column of each data frame is used as the metaFeatureID, and thus should match the row names of any metaAssays added via `addMetaAssays`. To share a data frame across multiple models, use the modelID "default". All columns will be coerced to character strings. |
| `plots` | A nested list containing custom plotting functions and plot metadata. The input object is a 3-level nested list. The first, or top-level list element name(s) must match the study `modelID`(s). The second, or mid-level list element name(s) must match the names of the plotting function(s) defined in the current R session (see Details below for function construction requirements). The third, or bottom-level list provides metadata to categorize, display, and support each plot. The accepted fields are `displayName`, `description`, `plotType`, `models`, and `packages`. `displayName` sets the plot name in the app and the `description` field will display as a tool tip when hovering over plotting dropdown menus. The `plotType` field is a character vector that categorizes the plot by 1) the number of features it supports ("`singleFeature`" or "`multiFeature`"), 2) the number of test results used by the plotting function ("`singleTest`", "`multiTest`"), 3) if data from one or more models is used (add "`multiModel`" to specify that data from two or more models are used in the plot; otherwise the plot is assumed to reference only data within the model specified by the top-level list element name), and 4) if the plot is interactive (add "`plotly`" to specify interactive plots built using the plotly package; otherwise the plot is assumed to be static). e.g., `plotType = c("multiFeature", "multiTest", "plotly")`. If you do not specify the `plotType`, the plot will be designated as `plotType =` |

c("singleFeature", "singleTest"). The models field is an optional character vector that specifies the models that should be used by the app when invoking your custom plotting function. This field is set to 'all' by default and is only used when plotType includes "multiModel". If this field is not included the app will assume all models in the study should be used with your plotting function. If the plotting function requires additional packages beyond those attached by default to a fresh R session, these must be defined in the element packages. To share a plotting functions across multiple models, use the modelID "default". Alternatively, to share a plot across a specific subset of models, you can explicitly add the same plotting function to each model (option available as of OmicNavigator 1.16.0).

mapping Feature IDs from models. The input object is a list of named data frames. For each data frame, column names indicate model names while rows indicate featureIDs per model. Features with same index position across columns are treated as mapped across models. For each model, feature IDs must match feature IDs available in the results object of the respective model. 1:N relationships are allowed.

Mapping list elements are required to be named as 'default' or after a model name as provided in addModels(). If a single data frame is provided, this list element is recommended to be named 'default'. For multiple list elements, each with its own data frame, list elements should be named after model name(s) (a single element may still be named 'default'). In that case, when navigating in ON front-end (FE), mapping element related to the selected model in the FE will be used in multimodel plots. If a selected model in FE does not have a corresponding mapping list element, it may still use the mapping list element called 'default' if this is available.

E.g., if in a study there are models "transcriptomics" and "proteomics" and the user wants to create a plot based on data from both, a mapping list should be provided with addMapping(). In this case, the mapping list element may be named 'default'. This should contain a data frame with column names 'transcriptomics' and 'proteomics', where feature IDs that map across models are found in the same row.

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". |
| `metaAssays` | The metaAssays from the study. The input object is a list of data frames (one per model). The row names should correspond to the metaFeatureIDs (second column of data frame added via `addMetaFeatures`). 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". |
| `objects` | Any arbitrary R objects from the study. The input object is a list of objects (one per model). To share an object 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. |
| `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 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`, `addMapping`, `addBarcodes`, `addReports`, `addResultsLinkouts`, `addEnrichmentsLinkouts`, `addMetaFeaturesLinkouts`, `addMetaAssays`, `addObjects`, `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

```
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 (via `validateStudy`) |

## 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

```
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 annotationID |
| `quiet` | Suppress messages (default: `FALSE`) |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.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()`).

## See Also

`addAnnotations`

## getAssays *Get assays from a study*

### Description

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` | Character vector of library directories to search for study packages. If `NULL`, uses `.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()`).

### See Also

`addAssays`

## getBarcodeData *Get data for barcode and violin plots*

### Description

Get data for barcode and violin plots

### Usage

```
getBarcodeData(study, modelID, testID, annotationID, termID, 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 |
| `annotationID` | Filter by annotationID |
| `termID` | Filter by termID |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.libPaths`. |

## 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 |
| `lowest` | (numeric) The lowest differential statistic, rounded down 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*

---

## Description

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` | Character vector of library directories to search for study packages. If `NULL`, uses `.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()`).

### See Also

`addBarcodes`

---

## `getEnrichments` *Get enrichments from a study*

---

### Description

Get enrichments from a study

### Usage

```
getEnrichments(
  study,
  modelID = NULL,
  annotationID = 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 |
| `annotationID` | Filter by annotationID |
| `testID` | Filter by testID |
| `quiet` | Suppress messages (default: `FALSE`) |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.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()`).

## See Also

`addEnrichments`

---

`getEnrichmentsAnnotations` *Get the annotations for the enrichments of an installed OmicNavigator study*

---

## Description

This is the API endpoint the app uses to populate the dropdown menu in the Enrichment Analysis tab with the list of available annotations for the selected model and study.

## Usage

```
getEnrichmentsAnnotations(study, modelID, libraries = NULL)
```

## Arguments

| | |
|---|---|
| `study` | An OmicNavigator study. Only accepts name of installed study package. |
| `modelID` | The modelID selected by the user in the app |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.libPaths`. |

## Details

The annotations correspond to those used when adding the enrichments with `addEnrichments`. Any optional tooltips correspond to the descriptions added with `addAnnotations`.

## Value

A named list. The names are the identifiers to be displayed in the dropdown menu, and each list element is a single character vector with the description to be used as a tooltip in the app. If no custom description was provided by the user, the tooltip text is simply the identifier.

## See Also

`getEnrichmentsStudies`, `getEnrichmentsModels`, `addEnrichments`, `addAnnotations`

## getEnrichmentsIntersection *getEnrichmentsIntersection*

### Description

getEnrichmentsIntersection

### Usage

```
getEnrichmentsIntersection(
  study,
  modelID,
  annotationID,
  mustTests,
  notTests,
  sigValue,
  operator,
  type,
  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 annotationID |
| `mustTests` | The testIDs for which a featureID (or termID for enrichment) must pass the filters |
| `notTests` | The testIDs for which a featureID (or termID for enrichment) must **not** pass the filters. In other words, if a featureID passes the filter for a testID specified in notTests, that featureID is removed from the output |
| `sigValue` | The numeric significance value to use as a cutoff for each column |
| `operator` | The comparison operators for each column, e.g. `"<"` |
| `type` | Type of p-value: (`"nominal"` or `"adjusted"`) |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.libPaths`. |

### Value

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

### See Also

`getEnrichmentsTable`

---

## getEnrichmentsLinkouts    *Get enrichments table linkouts from a study*

---

### 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 annotationID |
| `quiet` | Suppress messages (default: `FALSE`) |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.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()`).

### See Also

`addEnrichmentsLinkouts`

## getEnrichmentsModels *Get the models for the enrichments of an installed OmicNavigator study*

### Description

This is the API endpoint the app uses to populate the dropdown menu in the Enrichment Analysis tab with the list of available models for the selected study.

### Usage

```
getEnrichmentsModels(study, libraries = NULL)
```

### Arguments

| | |
|---|---|
| `study` | An OmicNavigator study. Only accepts name of installed study package. |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.libPaths`. |

### Details

The models correspond to those used when adding the results with `addEnrichments`. Any optional tooltips correspond to the descriptions added with `addModels`.

### Value

A named list. The names are the identifiers to be displayed in the dropdown menu, and each list element is a single character vector with the description to be used as a tooltip in the app. If no custom description was provided by the user, the tooltip text is simply the identifier.

### See Also

`getEnrichmentsStudies`, `getResultsModels`, `addEnrichments`, `addModels`

## 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 `onStudy`, or the name of an installed study package. |
| `modelID` | Filter by modelID |
| `annotationID` | Filter by annotationID |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.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) |

## `getEnrichmentsStudies` *Get installed OmicNavigator studies that have enrichments*

### Description

This is the API endpoint the app uses to populate the dropdown menu in the Enrichment Analysis tab with the list of available studies with enrichments data.

### Usage

```
getEnrichmentsStudies(libraries = NULL)
```

### Arguments

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

### Details

Internally, `getEnrichmentsStudies` calls `getInstalledStudies` with `hasElements = "enrichments"`.

### Value

Returns a character vector of the installed OmicNavigator study packages

### See Also

`getInstalledStudies`, `getResultsStudies`

## `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 annotationID |
| `type` | Type of p-value: (`"nominal"` or `"adjusted"`) |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.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` *getEnrichmentsUpset*

### Description

getEnrichmentsUpset

### Usage

```
getEnrichmentsUpset(
  study,
  modelID,
  annotationID,
  sigValue,
  operator,
  type,
  tests = NULL,
  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 annotationID |
| `sigValue` | The numeric significance value to use as a cutoff for each column |
| `operator` | The comparison operators for each column, e.g. `"<"` |
| `type` | Type of p-value: (`"nominal"` or `"adjusted"`) |
| `tests` | Restrict UpSet plot to only include these tests |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.libPaths`. |

### 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.

### Usage

```
getFavicons(linkouts)
```

### Arguments

| | |
|---|---|
| `linkouts` | Character vector or (potentially nested) list of character vectors containing the URLs for the table linkouts. |

### Value

The URLs to the favicons for each linkout. The output returned will always be the same class and structure as the input.

### See Also

`getResultsLinkouts`, `getEnrichmentsLinkouts`

## `getFeatures` *Get features from a study*

### Description

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` | Character vector of library directories to search for study packages. If `NULL`, uses `.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.

### See Also

`addFeatures`

---

## `getInstalledStudies` *Get installed OmicNavigator studies*

---

### Description

Get installed OmicNavigator studies

### Usage

```
getInstalledStudies(hasElements = NULL, libraries = NULL)
```

### Arguments

| | |
|---|---|
| `hasElements` | Character vector of elements that must be present in the study packages. Valid elements are 'metaFeatures', 'results', 'enrichments', 'reports', 'plots', 'assays', 'samples', 'features', 'resultsLinkouts', and 'metaAssays'. If `NULL` (default), then all installed OmicNavigator studies are returned, regardless of their contents. |
| `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

### See Also

`getResultsStudies`, `getEnrichmentsStudies`

## `getLinkFeatures` *Get the shared features in a network link*

### Description

Get the shared features in a network link

### Usage

```
getLinkFeatures(study, annotationID, termID1, termID2, libraries = NULL)
```

### Arguments

| | |
|---|---|
| `study` | An OmicNavigator study. Only accepts name of installed study package. |
| `annotationID` | Filter by annotationID |
| `termID1, termID2` | Linked terms to find overlapping features |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.libPaths`. |

### Value

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

### See Also

`getNodeFeatures`

## `getMapping` *Get mapping object from a study*

### Description

Get mapping object from a study

### Usage

```
getMapping(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` | Character vector of library directories to search for study packages. If `NULL`, uses `.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()`).

## See Also

`addMapping`

---

## `getMetaAssays` *Get metaAssays from a study*

---

### Description

Get metaAssays from a study

### Usage

```
getMetaAssays(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` | Character vector of library directories to search for study packages. If `NULL`, uses `.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()`).

### See Also

`addMetaAssays`

## `getMetaFeatures` *Get metaFeatures from a study*

### Description

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` | Character vector of library directories to search for study packages. If `NULL`, uses `.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()`).

### See Also

`addMetaFeatures`

## `getMetaFeaturesLinkouts` *Get metaFeatures table linkouts from a study*

### Description

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` | Character vector of library directories to search for study packages. If `NULL`, uses `.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()`).

### See Also

`addMetaFeaturesLinkouts`

---

## `getMetaFeaturesTable` *Get metaFeatures for a given feature*

---

### Description

Get metaFeatures for a given feature

### Usage

```
getMetaFeaturesTable(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` | Character vector of library directories to search for study packages. If `NULL`, uses `.libPaths`. |

### 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` *Get models from a study*

---

## 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` | Character vector of library directories to search for study packages. If `NULL`, uses `.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()`).

## See Also

`addModels`

## getNodeFeatures *Get the features in a network node*

### Description

Get the features in a network node

### Usage

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

### Arguments

| | |
|---|---|
| `study` | An OmicNavigator study. Only accepts name of installed study package. |
| `annotationID` | Filter by annotationID |
| `termID` | Filter by termID |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.libPaths`. |

### Value

Returns a character vector with the features in the termID

### See Also

`getLinkFeatures`

## getObjects *Get objects from a study*

### Description

Get objects from a study

### Usage

```
getObjects(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` | Character vector of library directories to search for study packages. If `NULL`, uses `.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()`).

## See Also

`addObjects`

---

`getOverlaps` *Get overlaps from a study*

---

## Description

Get overlaps from a study

## Usage

```
getOverlaps(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 annotationID |
| `quiet` | Suppress messages (default: `FALSE`) |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.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()`).

## See Also

`addOverlaps`

## `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

```
getPackageVersion(libraries = NULL)
```

### Arguments

| | |
|---|---|
| `libraries` | Directory path(s) to R package library(ies). Passed to the argument `lib.loc` of `packageVersion`. |

### Value

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

### See Also

`packageVersion`

## `getPlots` *Get plots from a study*

### Description

Get plots from a study

### Usage

```
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` | Character vector of library directories to search for study packages. If `NULL`, uses `.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()`).

## See Also

`addPlots`

---

`getPlottingData` *Get plotting data from an OmicNavigator study*

---

## Description

Returns `assays`, `samples`, and `features` data that may be used for plotting. This function is called by `plotStudy()` and the output is passed to custom plotting functions. It should be used directly when interactively creating custom plotting functions. Optionally, it can also return data for `results`, `metaFeatures`, `metaAssays`.

## Usage

```
getPlottingData(study, modelID, featureID, testID = NULL, 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 |
| `testID` | Filter by testID |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.libPaths`. |

## Details

The end-user should call this function and populate the first argument of their custom plotting function with the output. When building functions, the end-user should understand the category of plotting function they are creating (e.g. `singleFeature` or `multiFeature`, see `addPlots()`) and call `getPlottingData()` accordingly.

Custom plots that accept data from multiple models and a single test (`plotType = c('multiModel', 'singleTest')`; see `addPlots()`) should be built to accept output from `getPlottingData()` where `modelID` is vector of length n and `testID` is a vector of length n, where n is the number of models. Custom plots that accept data from multiple models and multiple tests (`plotType = c('multiModel', 'multiTest')`) should be built to accept output from `getPlottingData()` where `modelID` and `testID` vectors are length m, where m is the total number of tests considered across all models (note that `testIDs` must be repeated across models for the plotting function to work in the app). The index positions of these two vectors should correspond. That is, `testID` position 1 should be found in the model specified by `modelID` position 1, etc. See `addPlots()` for information about the assignment of `plotTypes` for your custom plots.

## Value

Returns a list of at least 3 elements:

| | |
|---|---|
| `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). |

If a testID is passed, the data frame `results` is also returned (by default the app will always pass the currently selected testID):

| | |
|---|---|
| `results` | A data frame that contains the test results, filtered to only include the row(s) corresponding to the input featureID(s). If multiple featureIDs are requested, the rows are reordered to match the order of this input. The column order is unchanged. If multiple testIDs are provided, they are stored in a list object. |

If the study has metaAssays available that map to the input featureID(s), then `metaFeatures` and `metaAssays` are returned:

| | |
|---|---|
| `metaFeatures` | A data frame that contains the metaFeature metadata, filtered to only include the row(s) corresponding to the input featureID(s) (see `getMetaFeatures`). If multiple featureIDs are requested, the rows are reordered to match the order of this input (and thus match the order of the metaAssays data frame). |
| `metaAssays` | A data frame that contains the metaAssay measurements, filtered to only include the row(s) corresponding to the input featureID(s) (see `getMetaAssays`). If multiple featureIDs are requested, the rows are reordered to match the order of this input. The column order is unchanged. |

If the study has objects available that map to the input modelID(s), then `objects` is returned. It is not possible to filter by featureID(s) since the structure of the custom object is unknown (and thus will need to be filtered by the plotting function code).

| | |
|---|---|
| `objects` | A custom object that was added to the modelID (`addObjects`) |

If multiple models are passed, then the top-level elements correspond to the names of the modelIDs, and the above elements are each nested within their respective modelID. Furthermore, an additional top-level element `mapping` is returned:

| | |
|---|---|
| `mapping` | A data frame that contains the featureID(s) from each model. This is the filtered mapping object. |

## See Also

`addPlots`, `plotStudy`

---

## `getReportLink` *Get link to report*

---

### Description

Get link to report

### Usage

```
getReportLink(study, modelID, 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 |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.libPaths`. |

### 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` | Character vector of library directories to search for study packages. If `NULL`, uses `.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()`).

### See Also

`addReports`

## 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` | Character vector of library directories to search for study packages. If `NULL`, uses `.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()`).

## See Also

`addResults`

---

`getResultsIntersection` *getResultsIntersection*

---

## Description

getResultsIntersection

## Usage

```
getResultsIntersection(
  study,
  modelID,
  anchor,
  mustTests,
  notTests,
  sigValue,
  operator,
  column,
  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 |
| `anchor` | The primary testID to filter the results |
| `mustTests` | The testIDs for which a featureID (or termID for enrichment) must pass the filters |
| `notTests` | The testIDs for which a featureID (or termID for enrichment) must **not** pass the filters. In other words, if a featureID passes the filter for a testID specified in notTests, that featureID is removed from the output |
| `sigValue` | The numeric significance value to use as a cutoff for each column |
| `operator` | The comparison operators for each column, e.g. `"<"` |
| `column` | The columns to apply the filters |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.libPaths`. |

## 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

```
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` | Character vector of library directories to search for study packages. If `NULL`, uses `.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()`).

### See Also

`addResultsLinkouts`

## `getResultsModels` *Get the models for the results of an installed OmicNavigator study*

### Description

This is the API endpoint the app uses to populate the dropdown menu in the Differential Analysis tab with the list of available models for the selected study.

### Usage

```
getResultsModels(study, libraries = NULL)
```

### Arguments

| | |
|---|---|
| `study` | An OmicNavigator study. Only accepts name of installed study package. |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.libPaths`. |

### Details

The models correspond to those used when adding the results with `addResults`. Any optional tooltips correspond to the descriptions added with `addModels`.

### Value

A named list. The names are the identifiers to be displayed in the dropdown menu, and each list element is a single character vector with the description to be used as a tooltip in the app. If no custom description was provided by the user, the tooltip text is simply the identifier.

### See Also

`getResultsStudies`, `getEnrichmentsModels`, `addResults`, `addModels`

---

## `getResultsStudies` *Get installed OmicNavigator studies that have results*

---

### Description

This is the API endpoint the app uses to populate the dropdown menu in the Differential Analysis tab with the list of available studies with results data.

### Usage

```
getResultsStudies(libraries = NULL)
```

### Arguments

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

### Details

Internally, `getResultsStudies` calls `getInstalledStudies` with `hasElements = "results"`.

### Value

Returns a character vector of the installed OmicNavigator study packages

### See Also

`getInstalledStudies`, `getEnrichmentsStudies` `getResultsTable` *Get results table from a study*

## Description

Get results table from a study

## Usage

```
getResultsTable(
  study,
  modelID,
  testID,
  annotationID = NULL,
  termID = NULL,
  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 |
| `annotationID` | Filter by annotationID |
| `termID` | Filter by termID |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.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.

If the optional arguments `annotationID` and `termID` are provided, the table will be filtered to only include features in that annotation term.

## `getResultsTests` *Get the tests for the results of an installed OmicNavigator study*

### Description

This is the API endpoint the app uses to populate the dropdown menu in the Differential Analysis tab with the list of available tests for the selected model and study.

### Usage

```
getResultsTests(study, modelID, libraries = NULL)
```

### Arguments

| | |
|---|---|
| `study` | An OmicNavigator study. Only accepts name of installed study package. |
| `modelID` | The modelID selected by the user in the app |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.libPaths`. |

### Details

The tests correspond to those used when adding the results with `addResults`. Any optional tooltips correspond to the descriptions added with `addTests`.

### Value

A named list. The names are the identifiers to be displayed in the dropdown menu, and each list element is a single character vector with the description to be used as a tooltip in the app. If no custom description was provided by the user, the tooltip text is simply the identifier.

### See Also

`getResultsStudies`, `getResultsModels`, `addResults`, `addTests`

## `getResultsUpset` *getResultsUpset*

### Description

getResultsUpset

### Usage

```
getResultsUpset(
  study,
  modelID,
  sigValue,
  operator,
  column,
  legacy = 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 |
| `sigValue` | The numeric significance value to use as a cutoff for each column |
| `operator` | The comparison operators for each column, e.g. `"<"` |
| `column` | The columns to apply the filters |
| `legacy` | Use legacy code (for testing purposes only) |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.libPaths`. |

### Value

Invisibly returns the output from `upset`

---

## `getSamples` *Get samples from a study*

---

### Description

Get samples from a study

### Usage

```
getSamples(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` | Character vector of library directories to search for study packages. If `NULL`, uses `.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()`).

## See Also

`addSamples`

---

`getStudyMeta` *Get study metadata*

---

## Description

Get the study description, version, maintainer, maintainer email, and any extra metadata added via the argument `studyMeta` of `createStudy`.

## Usage

```
getStudyMeta(study, libraries = NULL)
```

## Arguments

| | |
|---|---|
| `study` | Name of an installed OmicNavigator study |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.libPaths`. |

## Value

Returns a list with the following components:

| | |
|---|---|
| `description` | (character) Study description |
| `version` | (character) Study version |
| `maintainer` | (character) Study maintainer |
| `maintainerEmail` | (character) Study maintainer email |
| `studyMeta` | (list) Additional study metadata added via the argument `studyMeta` of `createStudy`) |

## See Also

`createStudy`

## `getTests` *Get tests from a study*

### Description

Get tests from a study

### Usage

```
getTests(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` | Character vector of library directories to search for study packages. If `NULL`, uses `.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()`).

### See Also

`addTests`

## `getUpsetCols` *getUpsetCols*

### Description

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

### Usage

```
getUpsetCols(study, modelID, 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 |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.libPaths`. |

## Value

Returns a character vector with the names of the common columns

---

`group` *group from Bioconductor workflow RNAseq123*

---

## 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

`https://bioconductor.org/packages/release/workflows/vignettes/RNAseq123/inst/doc/limmaWorkflow.html`

## References

Law CW, Alhamdoosh M, Su S, Dong X, Tian L, Smyth GK, Ritchie ME. RNA-seq analysis is easy as 1-2-3 with limma, Glimma and edgeR [version 3; peer review: 3 approved]. F1000Research 2018, 5:1408 doi:10.12688/f1000research.9005.3

Sheridan, J.M., Ritchie, M.E., Best, S.A. et al. A pooled shRNA screen for regulators of primary mammary stem and progenitor cells identifies roles for *Asap1* and *Prox1*. BMC Cancer 2015, 15:221 doi:10.1186/s128850151187z

## Examples

```
table(group)
str(group)
```

## `importStudy` *Import a study package*

### Description

Create an onStudy object by importing an installed study package

### Usage

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

### Arguments

| | |
|---|---|
| `study` | Name of an installed OmicNavigator study |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.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

```
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"`. |

## Value

A one-element character vector with the absolute path to the directory in which the app files were installed

---

`installStudy` *Install a study as an R package*

---

## Description

Install a study as an R package

## Usage

```
installStudy(study, requireValid = TRUE, library = .libPaths()[1])
```

## Arguments

| | |
|---|---|
| `study` | An OmicNavigator study to install (class `onStudy`) |
| `requireValid` | Require that study is valid before installing (passed to `exportStudy`, which runs `validateStudy`) |
| `library` | Directory to install package. Defaults to first directory returned by `.libPaths`. |

## Details

Note that `installStudy` is only intended for directly installing an OmicNavigator study object loaded in your current R session. If you have already exported your study to a package tarball via `exportStudy`, then you can install it with `install.packages`, for example:

```
tarball <- exportStudy(myStudy)
install.packages(tarball, repos = NULL)
```

## Value

Invisibly returns the original `onStudy` object that was passed to the argument `study` `lane` *lane from Bioconductor workflow RNAseq123*

## 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

## Source

`https://bioconductor.org/packages/release/workflows/vignettes/RNAseq123/inst/doc/limmaWorkflow.html`

## References

Law CW, Alhamdoosh M, Su S, Dong X, Tian L, Smyth GK, Ritchie ME. RNA-seq analysis is easy as 1-2-3 with limma, Glimma and edgeR [version 3; peer review: 3 approved]. F1000Research 2018, 5:1408 doi:10.12688/f1000research.9005.3

Sheridan, J.M., Ritchie, M.E., Best, S.A. et al. A pooled shRNA screen for regulators of primary mammary stem and progenitor cells identifies roles for *Asap1* and *Prox1*. BMC Cancer 2015, 15:221 doi:10.1186/s128850151187z

## Examples

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

## `lcpm` *lcpm from Bioconductor workflow RNAseq123*

### Description

A subset of the object `lcpm` from Bioconductor workflow RNAseq123.

### Usage

```
lcpm
```

### Format

A matrix with 24 rows and 9 columns

### Source

`https://bioconductor.org/packages/release/workflows/vignettes/RNAseq123/inst/doc/limmaWorkflow.html`

### References

Law CW, Alhamdoosh M, Su S, Dong X, Tian L, Smyth GK, Ritchie ME. RNA-seq analysis is easy as 1-2-3 with limma, Glimma and edgeR [version 3; peer review: 3 approved]. F1000Research 2018, 5:1408 doi:10.12688/f1000research.9005.3

Sheridan, J.M., Ritchie, M.E., Best, S.A. et al. A pooled shRNA screen for regulators of primary mammary stem and progenitor cells identifies roles for *Asap1* and *Prox1*. BMC Cancer 2015, 15:221 doi:10.1186/s128850151187z

### Examples

```
head(lcpm)
str(lcpm)
```

## `Mm.c2` *Mm.c2 from Bioconductor workflow RNAseq123*

### Description

A subset of the object `Mm.c2` from Bioconductor workflow RNAseq123.

### Usage

```
Mm.c2
```

### Format

A list of 4 character vectors

### Source

`https://bioconductor.org/packages/release/workflows/vignettes/RNAseq123/inst/doc/limmaWorkflow.html`

### References

Law CW, Alhamdoosh M, Su S, Dong X, Tian L, Smyth GK, Ritchie ME. RNA-seq analysis is easy as 1-2-3 with limma, Glimma and edgeR [version 3; peer review: 3 approved]. F1000Research 2018, 5:1408 doi:10.12688/f1000research.9005.3

Sheridan, J.M., Ritchie, M.E., Best, S.A. et al. A pooled shRNA screen for regulators of primary mammary stem and progenitor cells identifies roles for *Asap1* and *Prox1*. BMC Cancer 2015, 15:221 doi:10.1186/s128850151187z

### Examples

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

---

## `plotStudy` *Invoke a custom plotting function*

---

### Description

`plotStudy()` invokes a custom plotting function saved within an OmicNavigator study. This function is called by the app using the study-model-test selection, feature selections, and plotting function metadata (see `addPlots()`) to define arguments.

### Usage

```
plotStudy(study, modelID, featureID, plotID, testID = NULL, 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 |
| `testID` | Filter by testID |
| `libraries` | Character vector of library directories to search for study packages. If `NULL`, uses `.libPaths`. |

### Details

The arguments `study`, `modelID`, `featureID`, and `testID` are passed to the function `getPlottingData()`. The list returned by `getPlottingData()` is passed as the first argument to a custom plotting function. Some custom `plotTypes` (see `addPlots()`) require care when being invoked and attention should be paid to how a custom plot will be rendered by the app. Custom plots with `plotType = c('multiModel', 'singleTest')` accept a `modelID` vector of length n and a vector of `testIDs` length n, where n is the number of models. Custom plots with `plotType = c('multiModel', 'multiTest')` accept `modelID` and `testID` vectors of length m, where m is the total number of tests considered across all models (note `testIDs` are often repeated across models). Note that the index positions of these two vectors should correspond. That is, `testID` position 1 should be found in the model specified by `modelID` position 1, etc.

The app will invoke custom plotting functions via `plotStudy()` using the current menu selections and plot metadata (see `addPlots()`). Plots with `plotType = 'multiTest'` will be invoked with all `testIDs` found within the currently selected model. Plots with `plotType = c('multiModel','singleTest')` will be invoked with all `modelIDs` within the study (unless the plot has specified a list of models via `models`) and the currently selected `testID` (an error will result if the currently selected `testID` is not present in all relevant models for the plot). Plots with `plotType = c('multiModel', 'multiTest')` will be invoked with all `modelIDs` within the study (unless the plot has specified a list of models via `models`) and all identical `testIDs` across models (if there are no matching testIDs across models an error will result).

### Value

This function is called for the side effect of creating a plot. It invisibly returns the result from the custom plotting function specified by `plotID`. Previously it invisibly returned the study object. It's unlikely you relied on this behavior. For a ggplot2 plot, the return value will be the plotting object with class `"ggplot"`. For a plotly plot, the return value will be the json schema used for plotting with class “`json`”.

### See Also

`addPlots`, `getPlottingData`

---

## `removeStudy` *Remove an installed study R package*

---

### Description

Remove an installed study R package

### Usage

```
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

`https://bioconductor.org/packages/release/workflows/vignettes/RNAseq123/inst/doc/limmaWorkflow.html`

### References

Law CW, Alhamdoosh M, Su S, Dong X, Tian L, Smyth GK, Ritchie ME. RNA-seq analysis is easy as 1-2-3 with limma, Glimma and edgeR [version 3; peer review: 3 approved]. F1000Research 2018, 5:1408 doi:10.12688/f1000research.9005.3

Sheridan, J.M., Ritchie, M.E., Best, S.A. et al. A pooled shRNA screen for regulators of primary mammary stem and progenitor cells identifies roles for *Asap1* and *Prox1*. BMC Cancer 2015, 15:221 doi:10.1186/s128850151187z

### 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

```
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:

```
install.packages(c("faviconPlease", "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

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

## 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` *Validate a 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