Package 'CytobankAPI'

July 21, 2025

Title Cytobank API Wrapper for R

Version 2.2.1

Description Tools to interface with Cytobank's API via R, organized by endpoints that represent various areas of Cytobank functionality. Learn more about Cytobank at <https://www.beckman.com/flow-cytometry/software>.

License GPL-3

Depends R (>= 3.5.0), curl (>= 2.7), httr (>= 1.2.1)

Imports jsonlite, methods, stats, aws.s3, uuid, jose

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

RoxygenNote 7.2.1

Encoding UTF-8

Copyright 2020 Beckman Coulter, Inc.

Config/testthat/edition 3

NeedsCompilation no

Author Stu Blair [aut, cre],
Qihao Qi [aut],
Stefanie Trop [aut],
Louis Liu [aut],
Preston Ng [aut],
Chris Ciccolella [aut],
Katherine Drake [aut]

Maintainer Stu Blair <sblair@beckman.com>

Repository CRAN

Date/Publication 2023-04-21 08:32:30 UTC

Contents

AdvancedAnalysis-class 2
attachments 3

authentication . . . 6
autogating . . . 7
citrus . . . 13
CITRUS-class . . . 16
compensations . . . 17
DimensionalityReduction-class . . . 19
dimensionality_reduction . . . 20
drop . . . 23
experiments . . . 24
fcs_files . . . 29
flowsom . . . 33
FlowSOM-class . . . 36
gates . . . 38
helper_functions . . . 40
news . . . 41
optSNE-class . . . 42
panels . . . 43
peacoqc . . . 44
PeacoQC-class . . . 47
populations . . . 48
sample_tags . . . 49
scales . . . 51
spade . . . 52
SPADE-class . . . 58
statistics . . . 59
tSNE-class . . . 61
UMAP-class . . . 62
users . . . 63
UserSession-class . . . 64
visne . . . 65
viSNE-class . . . 68

Index **70**

`AdvancedAnalysis-class` *S4 Advanced Analysis Class*

Description

An Advanced Analysis object that is a parent class to all advanced analysis algorithms. This class should never be called explicitly. Its purpose is to act as a parent class for advanced analyses.

Value

An Advanced Analysis object

Slots

`channels` the channels selected for the advanced analysis, this can be either a list of short channel IDs (integer) OR long channel names (character)

`compensation_id` the compensation ID selected for the advanced analysis

`name` the name of the advanced analysis

`source_experiment` the source experiment ID the advanced analysis is associated with

`status` character representing the status of the advanced analysis

`.available_channels` the list of available channels based off the panels.list function

`.available_files` the list of available files based off the fcs_files.list function

`.available_populations` the list of available populations based off the populations.list function

---

`attachments` *Attachment Endpoints*

Description

Interact with attachments using these endpoints. Only FCS files can be analyzed in Cytobank, but any file can be uploaded as an attachment. Exported PDFs, statistics, and files also automatically attach themselves to the Experiment they are exported from. Learn more about attachments in Cytobank.

Usage

```
## S4 method for signature 'UserSession'
attachments.delete(
  UserSession,
  experiment_id,
  attachment_id,
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
attachments.download(
  UserSession,
  experiment_id,
  attachment_id,
  directory = getwd(),
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
attachments.download_zip(
  UserSession,
  experiment_id,
```

```
  attachment_id,
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
attachments.list(
  UserSession,
  experiment_id,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
attachments.show(
  UserSession,
  experiment_id,
  attachment_id,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
attachments.update(
  UserSession,
  attachment,
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
attachments.upload(
  UserSession,
  experiment_id,
  file_path,
  output = "default",
  timeout = UserSession@long_timeout
)
```

Arguments

`UserSession`	Cytobank UserSession object
`experiment_id`	integer representing an experiment ID
`attachment_id`	integer representing an attachment ID
`timeout`	integer representing the request timeout time in seconds **[optional]**
`directory`	character representing a specific directory to which the file will be downloaded (optional ending directory slash), if left empty, the default will be the current working directory **[optional]**
`output`	character representing the output format **[optional]** *- attachments.list, attachments.show, attachments.update :* `("default", "raw")`

attachment	dataframe representing an attachment (can retrieve via the attachments.show endpoint)
file_path	character representing a file path

Details

attachments.delete Permanently delete an attachment.

attachments.download Download an attachment from an experiment.

attachments.download_zip Download all or a select set of attachments as a zip file from an experiment. The download link of the zip file will be sent to the user's registered email address.

attachments.list List all attachments from an experiment. Outputs a dataframe [default] or raw list with all fields present.
- *Optional output parameter, specify one of the following:* ("default", "raw")

attachments.show Show attachment details from an experiment.
- *Optional output parameter, specify one of the following:* ("default", "raw")

attachments.update Update an attachment description from an experiment.

attachments.upload Upload an attachment to an experiment.
- *Optional output parameter, specify one of the following:* ("default", "raw")

Examples

```
## Not run: # Authenticate via username/password
cyto_session <- authenticate(site="premium", username="cyril_cytometry", password="cytobank_rocks!")
Authenticate via auth_token
cyto_session <- authenticate(site="premium", auth_token="my_secret_auth_token")

## End(Not run)
## Not run: attachments.delete(cyto_session, 22, attachment_id=2)

## Not run: # Download an attachment to the current working directory
attachments.download(cyto_session, 22)

# Download an attachment to a new directory
attachments.download(cyto_session, 22, directory="/my/new/download/directory/")

## End(Not run)
## Not run: # Download the all attachment files as a zip file
attachments.download_zip(cyto_session, experiment_id=22)

# Download a select set of attachment files as a zip file
attachments.download_zip(cyto_session, experiment_id=22, attachment_id=2)

## End(Not run)
## Not run: # Dataframe of all attachments with all fields present
attachments.list(cyto_session, 22)

# Raw list of all attachments with all fields present
attachments.list(cyto_session, 22, output="raw")
```

```
## End(Not run)
## Not run: attachments.show(cyto_session, 22, attachment_id=2)

## Not run: attachments.update(cyto_session, attachment=cyto_attachment)

## Not run: attachments.upload(cyto_session, 22, file_path="/path/to/my_attachment.txt")
```

`authentication` *Authentication Endpoints*

Description

Interact with authentication endpoints. Every call to the Cytobank API must be accompanied by an authentication token. Tokens should be kept secure as they confer access to the data and analyses of an account. Tokens expire after 8 hours by default but this figure my change depending on custom configurations of an Enterprise Cytobank.

Usage

```
authenticate(
  site,
  username = NA,
  password = NA,
  auth_token = NA,
  short_timeout = 30,
  long_timeout = 60,
  timeout = 30
)

## S4 method for signature 'UserSession'
authentication.logout(UserSession, timeout = UserSession@short_timeout)

## S4 method for signature 'UserSession'
authentication.revoke_all_tokens(
  UserSession,
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
authentication.revoke_all_tokens_user(
  UserSession,
  user_id,
  timeout = UserSession@short_timeout
)
```

Arguments

site	character representing Cytobank user's site, as in 'site'.cytobank.org. If your Cytobank server does not end in '.org', enter the entire server name, as in 'site.cytobank.cn'.
username	character representing Cytobank user's username or email
password	character representing Cytobank user's password
auth_token	character representing Cytobank user's authentication token (expires in 8 hours)
short_timeout	numeric representing short request timeout times (default = 30s) **[optional]**
long_timeout	numeric representing long request timeout times (default = 60s) **[optional]**
timeout	integer representing the request timeout time in seconds **[optional]**
UserSession	Cytobank UserSession object
user_id	integer representing Cytobank user's ID

Details

`authenticate` Authenticate a Cytobank user and return a Cytobank UserSession object that is passed to all other Cytobank API endpoints.

`authentication.logout` This function has been deprecated. Logout a Cytobank user.

`authentication.revoke_all_tokens` This function has been deprecated. Invalidate all existing tokens for the user making this call.

`authentication.revoke_all_tokens_user` This function has been deprecated. Revoke all tokens for a given user. This endpoint only works for admins of the Cytobank site being accessed.

Examples

```
## Not run: # Authenticate via username/password
cyto_session <- authenticate(site="premium", username="cyril_cytometry",
password="cytobank_rocks!") # Authenticate via auth_token
cyto_session <- authenticate(site="premium", auth_token="my_secret_auth_token")

## End(Not run)
```

`autogating` *Automatic gating Endpoints*

Description

Automatic gating Endpoints

Usage

```
## S4 method for signature 'UserSession'
autogating.list_autogating_analyses_of_type(
  UserSession,
  experiment_id,
  analysis_type,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
autogating.show_autogating_analysis_details(
  UserSession,
  experiment_id,
  analysis_id,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
autogating.create_autogating_analysis(
  UserSession,
  experiment_id,
  analysis_type,
  name,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
autogating.update_autogating_training_analysis_details(
  UserSession,
  experiment_id,
  analysis_id,
  createBlindTestExperiment,
  desiredEventsPerFile,
  desiredTotalEvents,
  eventSamplingMethod,
  fcsFileIds,
  gateSetIds,
  learningMagnification,
  optimalClusters,
  randomSeed,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
```

```
autogating.update_autogating_inference_analysis_details(
  UserSession,
  experiment_id,
  analysis_id,
  cloneGatesFromParent,
  fcsFileIds,
  trainedModelAnalysisId,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
autogating.delete_autogating_analysis(
  UserSession,
  experiment_id,
  analysis_id,
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
autogating.copy_autogating_analysis_settings(
  UserSession,
  experiment_id,
  analysis_id,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
autogating.rename_autogating_analysis(
  UserSession,
  experiment_id,
  analysis_id,
  name,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
autogating.run_autogating_analysis(
  UserSession,
  experiment_id,
  analysis_id,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
```

```
autogating.show_autogating_analysis_status(
  UserSession,
  experiment_id,
  analysis_id,
  output = "default",
  timeout = UserSession@long_timeout
)
```

Arguments

`UserSession`	Cytobank UserSession object
`experiment_id`	integer representing an experiment ID
`analysis_type`	character representing the type of ananlysis: auto_gate_train or auto_gate_inference
`output`	character representing the output format **[optional]** - *drop.upload :* `("default", "raw")` - `dataframe`*: converts the file internal compensation matrix output to a dataframe*
`timeout`	integer representing the request timeout time in seconds **[optional]**
`analysis_id`	integer representing the id of an autogating analysis
`name`	character representing the name of an autogating analysis
`createBlindTestExperiment`	boolean A child experiment will be automatically created, containing the subset of FCS files that were assigned to the blind test set. For every predicted population, the files now contain one additional parameter following the naming convention of auto_gate_Population name.
`desiredEventsPerFile`	integer Only applies if eventSamplingMethod is set to equal. Defaults to 50,000. It is the number of desired events to sample per file, but if the selected population for any selected file has less total events than the specified number, that quantity will be used instead.
`desiredTotalEvents`	integer Only applies if eventSamplingMethod is set to proportional. Defaults to 5,000,000. Represents the total desired number of events to sample amongst all selected files, whilst keeping the numbers per file proportional to the total number of events in the selected population for that file. If any file has less events in the selected population than possible to make a perfectly proportional sampling to add up to the desired total, all of the events in the file will be used instead.
`eventSamplingMethod`	character Valid options are proportional, equal, or all. Defaults to equal. If eventSamplingMethod is set to all, all events for the selected population from all selected files will be used, without any further subsampling.
`fcsFileIds`	vector/list representing the id list of FCS files
`gateSetIds`	vector/list representing the id list of Cytobank gate set
`learningMagnification`	integer By increasing the magnification, the user can determine how many different models are being trained using different parameters on the same training

data. The model with the highest KPI will be returned to the user. With a magnification greater than 1, you may be able to influence the model selection to return a model performing better on your population of interest, but usually not significant. Of note, increasing the magnification also causes a proportional increase of the runtime. It may also cause the run to crash due to memory constraints if there are millions of events.

`optimalClusters`
: integer The best estimate of the number of distinct groups of files amongst those selected. Usually, this aligns with how you would sample tag your files into different conditions or time points. It helps the algorithm pick representative samples and perform better. There is an option to create an experiment with blind test files and their inferred populations. It can make it easier to visually evaluate model performance.

`randomSeed`
: integer Accepts a positive integer value and sets a specific random seed to that value. If this parameter is not specified or set to 0, autoSeed will automatically be set to true, and a seed value will be randomly chosen, so that afterward it can be referred to for reproducing the analysis results.

`cloneGatesFromParent`
: boolean The created child experiment will contain a copy of all gates & populations already present in the parent experiment

`trainedModelAnalysisId`
: character The ID of the Autogate Training analysis that contains the model that the inference run will use.

Details

`autogating.list_autogating_analyses_of_type`

`autogating.show_autogating_analysis_details`

`autogating.create_autogating_analysis` Create a new automatic gating analysis of the specified type (auto_gate_train or auto_gate_inference).

`autogating.update_autogating_training_analysis_details`

`autogating.update_autogating_inference_analysis_details`

`autogating.delete_autogating_analysis`

`autogating.copy_autogating_analysis_settings`

`autogating.rename_autogating_analysis`

`autogating.run_autogating_analysis`

`autogating.show_autogating_analysis_status`

Examples

```
## Not run:
# Create train analysis
autogating_train_analysis <- autogating.list_autogating_analyses_of_type(cyto_session,
                             p_experiment_id,
                             "auto_gate_train")
# Update train settings
```

```
autogating.update_autogating_training_analysis_details(cyto_session, p_experiment_id,
                    autogating_train_analysis$id,
                    FALSE, 39139, 100001, "proportional",
               c(114386,114373,114383,114374,114384,114387,114385,114377,114382,114375),
                    c(4,3,1,11,10), 1, 2, 1)
# Run analysis
autogating.run_autogating_analysis(cyto_session, p_experiment_id, autogating_train_analysis$id)
# Create inference analysis
autogating_inference_analysis <- autogating.list_autogating_analyses_of_type(cyto_session,
                                          p_experiment_id,
                                          "auto_gate_inference")
# Update inference settings
autogating.update_autogating_inference_analysis_details(cyto_session, p_experiment_id,
                                  autogating_inference_analysis$id, FALSE,
                             c(114376,114378,114379,114380,114381,114388,114389,114390),
                                  autogating_train_analysis$id)
# Run analysis
autogating.run_autogating_analysis(cyto_session, p_experiment_id,
                                 autogating_inference_analysis$id)

## End(Not run)
## Not run: autogating.list_autogating_analyses_of_type(cyto_session, 22, "auto_gate_train")

## Not run: autogating.show_autogating_analysis_details(cyto_session, 22, 10)

## Not run: autogating.create_autogating_analysis(cyto_session, 22, "auto_gate_train",
                                                        "My auto gating train analysis")

## End(Not run)
## Not run: autogating.update_autogating_training_analysis_details(
                                       cyto_session, 22, 10, FALSE, 5000, 100000,
                                       "proportional",
                                       c(10, 11, 12, 13, 14, 15, 16, 17, 18, 19),
                                       c(3, 4), 1, 2, NULL)

## End(Not run)
## Not run: autogating.update_autogating_inference_analysis_details(
                                    cyto_session, 22, 10, FALSE, c(21, 22, 23), 10)

## End(Not run)
## Not run: autogating.delete_autogating_analysis(cyto_session, 22, 10)

## Not run: autogating.copy_autogating_analysis_settings(cyto_session, 22, 10)

## Not run: autogating.rename_autogating_analysis(cyto_session, 22, 10, "New new of analysis")

## Not run: autogating.run_autogating_analysis(cyto_session, 22, 10)

## Not run: autogating.show_autogating_analysis_status(cyto_session, 22, 10)
```

citrus *CITRUS Endpoints*

Description

Interact with CITRUS advanced analyses using these endpoints.

Usage

```
## S4 method for signature 'UserSession,CITRUS'
citrus.copy_settings(
  UserSession,
  citrus,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession,CITRUS'
citrus.delete(UserSession, citrus, timeout = UserSession@short_timeout)

## S4 method for signature 'UserSession,CITRUS'
citrus.download(
  UserSession,
  citrus,
  directory = getwd(),
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
citrus.list(
  UserSession,
  experiment_id,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
citrus.new(
  UserSession,
  experiment_id,
  citrus_name,
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession,CITRUS'
citrus.rename(
  UserSession,
```

```
  citrus,
  citrus_name,
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession,CITRUS'
citrus.run(
  UserSession,
  citrus,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
citrus.show(
  UserSession,
  experiment_id,
  citrus_id,
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession,CITRUS'
citrus.status(
  UserSession,
  citrus,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession,CITRUS'
citrus.update(UserSession, citrus, timeout = UserSession@long_timeout)
```

Arguments

`UserSession`	Cytobank UserSession object
`citrus`	Cytobank CITRUS object
`output`	character representing the output format **[optional]** - *citrus.list, citrus.run, citrus.status :* `("default", "raw")`
`timeout`	integer representing the request timeout time in seconds **[optional]**
`directory`	character representing a specific directory to which the file will be downloaded (optional ending directory slash), if left empty, the default will be the current working directory **[optional]**
`experiment_id`	integer representing an experiment ID
`citrus_name`	character representing a new CITRUS name
`citrus_id`	integer representing a CITRUS ID

Details

`citrus.copy_settings` Copy CITRUS advanced analysis settings from an experiment and returns a CITRUS object.

`citrus.delete` Delete a CITRUS advanced analysis from an experiment.

`citrus.download` Download a CITRUS analysis from an experiment.

`citrus.list` List all CITRUS advanced analyses from an experiment. Outputs a dataframe [default] or list with all fields present.
- *Optional output parameter, specify one of the following:* `("default", "raw")`

`citrus.new` Create a new CITRUS advanced analysis from an experiment and returns a CITRUS object.

`citrus.rename` Rename a CITRUS advanced analysis from an experiment and returns a CITRUS object.

`citrus.run` Run a CITRUS advanced analysis from an experiment.

`citrus.show` Show CITRUS advanced analysis details from an experiment and returns a CITRUS object.

`citrus.status` Show the status of a CITRUS advanced analysis from an experiment.

`citrus.update` Update a CITRUS advanced analysis from an experiment and returns the new CITRUS object.

Examples

```
## Not run: # Authenticate via username/password
cyto_session <- authenticate(site="premium", username="cyril_cytometry", password="cytobank_rocks!")
# Authenticate via auth_token
cyto_session <- authenticate(site="premium", auth_token="my_secret_auth_token")

# cyto_citrus refers to a CITRUS object that is created from CITRUS endpoints
#   examples: citrus.new, citrus.show (see details section for more)

## End(Not run)
## Not run: citrus.copy_settings(cyto_session, citrus=cyto_citrus)

## Not run: citrus.delete(cyto_session, citrus=cyto_citrus)

## Not run: # Download a CITRUS analysis to the current working directory
citrus.download(cyto_session, citrus)

# Download a CITRUS analysis to a new directory
citrus.download(cyto_session, citrus, directory="/my/new/download/directory/")

## End(Not run)
## Not run: # Dataframe of all CITRUS advanced analyses with all fields present
citrus.list(cyto_session, 22)

# Raw list of all CITRUS advanced analyses with all fields present
citrus.list(cyto_session, 22, output="raw")

## End(Not run)
```

```
## Not run: citrus.new(cyto_session, 22, citrus_name="My new CITRUS analysis")

## Not run: citrus.rename(cyto_session, citrus=cyto_citrus, citrus_name="My updated CITRUS name")

## Not run: citrus.run(cyto_session, citrus=cyto_citrus)

## Not run: citrus.show(cyto_session, 22, citrus_id=2)

## Not run: citrus.status(cyto_session, citrus=cyto_citrus)

## Not run: citrus.update(cyto_session, citrus=cyto_citrus)
```

`CITRUS-class` *S4 CITRUS Class*

Description

A CITRUS object that holds pertinent CITRUS advanced analysis run information, learn more about CITRUS. This class should never be called explicitly. If a user would like to create a new Cytobank CITRUS object, utilize the citrus.new function, or any other CITRUS endpoints that return CITRUS objects documented in the 'Details' section.

Value

A CITRUS advanced analysis object

Slots

`associated_models` list representing statistical methods used to discover stratifying signatures from clustered data features that explain differences between sample groups, learn more about CITRUS association models
- *choose from the following :* `("sam", "pamr" [default], "glmnet")`

`attachment_id` numeric representing the CITRUS attachment ID

`cross_validation_folds` numeric representing the regulation threshold, controlling the number of features in the model (only applies to PAM, LASSO), learn more about CITRUS cross validation folds

`citrus_id` numeric representing the CITRUS analysis ID

`cluster_characterization` character representing the principle for analyzing and quantifying individual samples, learn more about CITRUS cluster characterization
- *choose one of the following :* `("abundance" [default], "medians")`

`event_sampling_method` character representing the sampling method, learn more about CITRUS event sampling methods
- *choose one of the following :* `("equal" [default], "max-per-file")`

`events_per_file` numeric representing the number of events taken from each sample

`false_discovery_rate` numeric representing the false discovery rate (only applies to PAM, SAM), learn more about CITRUS false discovery rate

`file_grouping` numeric dataframe representing which group samples belong to, learn more about CITRUS file grouping, the core functionality of CITRUS

`minimum_cluster_size` numeric representing the number of nodes, learn more about CITRUS minimum cluster size

`normalize_scales` logical representing whether or not to normalize channels, learn more about normalizing CITRUS scales

`plot_theme` character representing the background color of images and figures within the CITRUS results
- *choose one of the following :* (`"white"` `[default]`, `"black"`)

`population_id` dataframe representing a population **gate set ID**

`statistics_channels` list representing the statistics channels used for the 'median' cluster characterization, these channels should not be selected for clustering

---

`compensations` *Compensation Endpoints*

Description

Interact with compensation endpoints. Get information about compensations stored in Cytobank. For information about file-internal compensation for an individual FCS file, consult the FCS files endpoints. Learn more about compensation in Cytobank.

Usage

```
## S4 method for signature 'UserSession'
compensations.upload_csv(
  UserSession,
  experiment_id,
  file_path,
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
compensations.list(
  UserSession,
  experiment_id,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
compensations.show(
  UserSession,
```

```
  experiment_id,
  compensation_id,
  output = "default",
  timeout = UserSession@short_timeout
)
```

Arguments

UserSession	Cytobank UserSession object
experiment_id	integer representing an experiment ID
file_path	character representing a file path
timeout	integer representing the request timeout time in seconds **[optional]**
output	character representing the output format **[optional]** - *compensations.list :* ("default", "raw") - *compensations.show :* ("default", "dataframe", "raw") - dataframe*: converts the compensation matrix output to a dataframe*
compensation_id	integer representing a compensation ID

Details

compensations.upload_csv Upload a compensation CSV to an experiment.

compensations.list List all compensations from an experiment. Outputs a formatted list [default] or raw list with all fields present.
- *Optional output parameter, specify one of the following:* ("default", "raw")

compensations.show Show compensation details from an experiment.
- *Optional output parameter, specify one of the following:* ("default", "dataframe", "raw")
- dataframe*: converts the compensation matrix output to a dataframe*

Examples

```
## Not run: # Authenticate via username/password
cyto_session <- authenticate(site="premium", username="cyril_cytometry", password="cytobank_rocks!")
# Authenticate via auth_token
cyto_session <- authenticate(site="premium", auth_token="my_secret_auth_token")

## End(Not run)
## Not run: compensations.upload_csv(cyto_session, 22, file_path="/path/to/my_compensation.csv")

## Not run: # List of all compensations with all fields present, with a compensation matrix dataframe list item
compensations.list(cyto_session, 22)

# Raw list of all compensations with all fields present
compensations.list(cyto_session, 22, output="raw")

## End(Not run)
## Not run: # List form of a compensation
compensations.show(cyto_session, 22, compensation_id=2)
```

```
# Compensation dataframe only
compensations.show(cyto_session, 22, compensation_id=2, output="dataframe")

## End(Not run)
```

`DimensionalityReduction-class`
S4 DimensionalityReduction Class

Description

A Dimensionality Reduction object that holds pertinent Dimensionality Reduction advanced analysis run information. This class should never be called explicitly. If a user would like to create a new Cytobank Dimensionality Reduction object, utilize the dimensionality_reduction.new function, or any other Dimensionality Reduction endpoints that return Dimensionality Reduction objects documented in the 'Details' section.

Value

A Dimensionality Reduction advanced analysis object

Slots

`analysis_id` numeric representing the Dimensionality Reduction analysis ID

`type` character representing the Dimensionality Reduction type (tSNE-CUDA, opt-SNE, UMAP, or viSNE)

`name` character the name of the Dimensionality Reduction analysis

`status` character representing the status of the Dimensionality Reduction analysis

`source_experiment` numeric the source experiment ID the Dimensionality Reduction analysis is associated with

`created_experiment` numeric representing the experiment that gets created from the Dimensionality Reduction analysis

`.available_channels` the list of available channels based off the panels.list function

`.available_files` the list of available files based off the fcs_files.list function

`.available_populations` the list of available populations based off the populations.list function

dimensionality_reduction *DimensionalityReduction Endpoints*

Description

Interact with DimensionalityReduction advanced analyses using these endpoints.

Usage

```
## S4 method for signature 'UserSession,DimensionalityReduction'
dimensionality_reduction.copy_settings(
  UserSession,
  dimensionality_reduction,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession,DimensionalityReduction'
dimensionality_reduction.delete(
  UserSession,
  dimensionality_reduction,
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
dimensionality_reduction.list(
  UserSession,
  experiment_id,
  analysis_type,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
dimensionality_reduction.new(
  UserSession,
  experiment_id,
  analysis_name,
  analysis_type,
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession,DimensionalityReduction'
dimensionality_reduction.rename(
  UserSession,
  dimensionality_reduction,
```

```
  analysis_name,
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession,DimensionalityReduction'
dimensionality_reduction.run(
  UserSession,
  dimensionality_reduction,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
dimensionality_reduction.show(
  UserSession,
  experiment_id,
  analysis_id,
  analysis_type,
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession,DimensionalityReduction'
dimensionality_reduction.status(
  UserSession,
  dimensionality_reduction,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession,DimensionalityReduction'
dimensionality_reduction.update(
  UserSession,
  dimensionality_reduction,
  timeout = UserSession@long_timeout
)
```

Arguments

`UserSession`	Cytobank UserSession object
`dimensionality_reduction`	Cytobank DimensionalityReduction object
`output`	character representing the output format **[optional]** *- dimensionality_reduction.list, dimensionality_reduction.run, dimensionality_reduction.status :* `("default", "raw")`
`timeout`	integer representing the request timeout time in seconds **[optional]**
`experiment_id`	integer representing an experiment ID
`analysis_type`	character representing the Dimensionality Reduction type (tSNE-CUDA, opt-SNE, UMAP, or viSNE)

analysis_name	character the name of the Dimensionality Reduction analysis
analysis_id	integer representing the Dimensionality Reduction analysis ID

Details

dimensionality_reduction.copy_settings Copy DimensionalityReduction advanced analysis settings from an experiment and returns a DimensionalityReduction object.

dimensionality_reduction.delete Delete a DimensionalityReduction advanced analysis from an experiment.

dimensionality_reduction.list List all DimensionalityReduction advanced analyses from an experiment. Outputs a dataframe [default] or list with all fields present.
- *Optional output parameter, specify one of the following:* ("default", "raw")

dimensionality_reduction.new Create a new DimensionalityReduction advanced analysis from an experiment and returns a DimensionalityReduction object.

dimensionality_reduction.rename Rename a DimensionalityReduction advanced analysis from an experiment and returns the new name.

dimensionality_reduction.run Run a DimensionalityReduction advanced analysis from an experiment.

dimensionality_reduction.show Show DimensionalityReduction advanced analysis details from an experiment and returns a DimensionalityReduction object.

dimensionality_reduction.status Show the status of a DimensionalityReduction advanced analysis from an experiment.

dimensionality_reduction.update Update a DimensionalityReduction advanced analysis from an experiment and returns the new DimensionalityReduction object.

Examples

```
## Not run: # Authenticate via username/password
cyto_session <- authenticate(site="premium", username="cyril_cytometry", password="cytobank_rocks!")
# Authenticate via auth_token
cyto_session <- authenticate(site="premium", auth_token="my_secret_auth_token")

# cyto_dimensionality_reduction refers to a DimensionalityReduction object that is created from
DimensionalityReduction endpoints
#  examples: dimensionality_reduction.new, dimensionality_reduction.show (see details section for
more)

## End(Not run)
## Not run: dimensionality_reduction.copy_settings(cyto_session,
 dimensionality_reduction=cyto_dimensionality_reduction)

## End(Not run)
## Not run: dimensionality_reduction.delete(cyto_session,
dimensionality_reduction=cyto_dimensionality_reduction)

## End(Not run)
## Not run: # Dataframe of all DimensionalityReduction advanced analyses with all fields present
dimensionality_reduction.list(cyto_session, 22, "viSNE")
```

```
# Raw list of all DimensionalityReduction advanced analyses with all fields present
dimensionality_reduction.list(cyto_session, 22, "viSNE", output="raw")

## End(Not run)
## Not run: dimensionality_reduction.new(cyto_session, 22,
analysis_name="My new DimensionalityReduction analysis", "UMAP")

## End(Not run)
## Not run: dimensionality_reduction.rename(cyto_session,
dimensionality_reduction=cyto_dimensionality_reduction,
analysis_name="My updated DimensionalityReduction name")

## End(Not run)
## Not run: dimensionality_reduction.run(cyto_session,
dimensionality_reduction=cyto_dimensionality_reduction)

## End(Not run)
## Not run: dimensionality_reduction.show(cyto_session, 22, analysis_id=2, "opt-SNE")

## Not run: dimensionality_reduction.status(cyto_session,
dimensionality_reduction=cyto_dimensionality_reduction)

## End(Not run)
## Not run: dimensionality_reduction.update(cyto_session,
dimensionality_reduction=cyto_dimensionality_reduction)

## End(Not run)
```

---

drop *DROP File Endpoints*

Description

Upload DROP file(s) into Cytobank. A DROP file consists of any CSV, TSV, TXT, or FCS file. If the DROP file is of the type CSV, TSV, or TXT, the file will be converted to an FCS file to be used within Cytobank. Learn more about DROP.

Usage

```
## S4 method for signature 'UserSession'
drop.upload(
  UserSession,
  experiment_id,
  file_path,
  data_matrix_start_row = 2,
  data_matrix_start_column = 1,
  skipped_columns = c(),
  output = "default",
```

```
  timeout = UserSession@long_timeout
)
```

Arguments

`UserSession`	Cytobank UserSession object
`experiment_id`	integer representing an experiment ID
`file_path`	character representing a file path
`data_matrix_start_row`	integer representing the start row of the DROP file(s)
`data_matrix_start_column`	integer representing the start column of the DROP file(s)
`skipped_columns`	vector/list of integer(s) representing column(s) of the DROP file to skip
`output`	character representing the output format **[optional]** - *drop.upload :* `("default", "raw")` - `dataframe`*: converts the file internal compensation matrix output to a dataframe*
`timeout`	integer representing the request timeout time in seconds **[optional]**

Details

`drop.upload` Upload a DROP file (CSV, TSV, TXT, FCS) to an experiment. - *Optional output parameter, specify one of the following:* `("default", "raw")`

Examples

```
## Not run: # Authenticate via username/password
cyto_session <- authenticate(site="premium", username="cyril_cytometry", password="cytobank_rocks!")
# Authenticate via auth_token
cyto_session <- authenticate(site="premium", auth_token="my_secret_auth_token")

## End(Not run)
## Not run: drop.upload(cyto_session, 22, file_path="/path/to/my_drop_file.type",
  data_matrix_start_row=2, data_matrix_start_column=1)

## End(Not run)
```

`experiments` *Experiment Endpoints*

Description

Interact with experiment endpoints. An Experiment is a container for data and analyses in Cytobank. If data are on Cytobank, they must be within an Experiment. Configurations such as gates, compensations, scales, Sample Tags, and illustrations are also linked to an individual Experiment. Within the Cytobank interface, the Experiment Summary Page is a useful integration point for information about an Experiment.

Usage

```
## S4 method for signature 'UserSession'
experiments.clone_full(
  UserSession,
  experiment_id,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
experiments.clone_selective(
  UserSession,
  experiment_id,
  experiment_name,
  fcs_files = c(-1),
  primary_researcher = NA,
  principal_investigator = NA,
  clone_gates = FALSE,
  clone_annotations = FALSE,
  clone_attachments = FALSE,
  clone_reagents = FALSE,
  clone_compensations = FALSE,
  clone_panels = FALSE,
  clone_illustrations = FALSE,
  clone_project = FALSE,
  clone_user_access = FALSE,
  allow_full_access_pi = FALSE,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
experiments.delete(
  UserSession,
  experiment_id,
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
experiments.full_access_users_list(
  UserSession,
  experiment_id,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
experiments.full_access_users_add(
```

```
  UserSession,
  experiment_id,
  user_id = NA,
  user_email = NA,
  username = NA,
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
experiments.full_access_users_remove(
  UserSession,
  experiment_id,
  user_id = NA,
  user_email = NA,
  username = NA,
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
experiments.list(
  UserSession,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
experiments.new(
  UserSession,
  experiment_name,
  purpose,
  comments = NA,
  primary_researcher = NA,
  principal_investigator = NA,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
experiments.show(
  UserSession,
  experiment_id,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
experiments.trash(
  UserSession,
```

```
  experiment_id,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
experiments.update(
  UserSession,
  experiment,
  output = "default",
  timeout = UserSession@short_timeout
)
```

Arguments

`UserSession`	Cytobank UserSession object
`experiment_id`	integer representing an experiment ID
`output`	character representing the output format **[optional]** *- experiments.clone_full, experiments.clone_selective, experiments.full_access_users_list, experiments.list, experiments.new, experiments.show, experiments.trash, experiments.update :* `("default", "raw")`
`timeout`	integer representing the request timeout time in seconds **[optional]**
`experiment_name`	character representing an experiment name
`fcs_files`	vector/list of integers representing a list of FCS file IDs **[optional]**
`primary_researcher`	integer representing a primary researcher ID **[optional]**
`principal_investigator`	integer representing a principal investigator ID **[optional]**
`clone_gates`	boolean denoting cloning gates option **[optional]**
`clone_annotations`	boolean denoting cloning annotations option **[optional]**
`clone_attachments`	boolean denoting cloning attachments option **[optional]**
`clone_reagents`	boolean denoting cloning reagents option **[optional]**
`clone_compensations`	boolean denoting cloning compensations option **[optional]**
`clone_panels`	boolean denoting cloning panels option **[optional]**
`clone_illustrations`	boolean denoting cloning illustrations option **[optional]**
`clone_project`	boolean denoting cloning project option **[optional]**
`clone_user_access`	boolean denoting cloning user access option **[optional]**
`allow_full_access_pi`	boolean denoting to allow full access to PI option **[optional]**

`user_id`	integer representing a user's ID
`user_email`	character representing a user's email
`username`	character representing a username
`purpose`	character representing an experiment purpose
`comments`	character representing an experiment comment **[optional]**
`experiment`	dataframe representing an experiment

Details

`experiments.clone_full` Full clone an experiment. Learn more about the full clone functionality.
- *Optional output parameter, specify one of the following:* `("default", "raw")`

`experiments.clone_selective` Selectively clone an experiment. Learn more about the selective clone functionality
- *Optional output parameter, specify one of the following:* `("default", "raw")`

`experiments.delete` Permanently delete an experiment and all analyses (including SPADE, viSNE, etc.) permanently. This is not reversible.

`experiments.list` List all full access users from an experiment.
- *Optional output parameter, specify one of the following:* `("default", "raw")`

`experiments.list` Add a full access user to an experiment. A full access user can be added by a user ID, email, or username.

`experiments.list` Remove a full access user from an experiment. A full access user can be removed by a user ID, email, or username.

`experiments.list` List all inbox experiments. Outputs a data frame [default] or raw list with all fields present.
- *Optional output parameter, specify one of the following:* `("default", "raw")`

`experiments.new` Create a new experiment.
- *Optional output parameter, specify one of the following:* `("default", "raw")`

`experiments.show` Show experiment details.
- *Optional output parameter, specify one of the following:* `("default", "raw")`

`experiments.trash` Trash an experiment. This is reversible and not to be confused with permanent deletion.

`experiments.update` Update an experiment. (all parameters are optional, except for experiment_id)
- *Optional output parameter, specify one of the following:* `("default", "raw")`

Examples

```
## Not run: # Authenticate via username/password
cyto_session <- authenticate(site="premium", username="cyril_cytometry", password="cytobank_rocks!")
# Authenticate via auth_token
cyto_session <- authenticate(site="premium", auth_token="my_secret_auth_token")

## End(Not run)
## Not run: experiments.clone_full(cyto_session, 22)

## Not run: experiments.clone_selective(cyto_session, 22,
```

```
  experiment_name="My New Experiment Name", fcs_files=c(12, 13, 14, 15, 16))

## End(Not run)
## Not run: experiments.delete(cyto_session, 22)

## Not run: # Dataframe of all full access users
experiments.full_access_users_list(cyto_session, 22)

# List of all full access users
experiments.full_access_users_list(cyto_session, 22, output="raw")

## End(Not run)
## Not run: # Add a user as a full access user by user's ID
experiments.full_access_users_add(cyto_session, 22, user_id=2)

# Add a user as a full access user by user's email
experiments.full_access_users_add(cyto_session, 22, user_email="sammy_cytometry@cytobank.org")

# Add a user as a full access user by user's username
experiments.full_access_users_add(cyto_session, 22, username="sammy_cytometry")

## End(Not run)
## Not run: # Remove a user as a full access user by user's ID
experiments.full_access_users_remove(cyto_session, 22, user_id=2)

# Remove a user as a full access user by user's email
experiments.full_access_users_remove(cyto_session, 22, user_email="sammy_cytometry@cytobank.org")

# Remove a user as a full access user by user's username
experiments.full_access_users_remove(cyto_session, 22, username="sammy_cytometry")

## End(Not run)
## Not run: # Dataframe of all inbox experiments with all fields present
experiments.list(cyto_session)

# Raw list of all inbox experiments with all fields present
experiments.list(cyto_session, output="raw")

## End(Not run)
## Not run: experiments.new(cyto_session, "My New Experiment Name", "My experiment purpose",
  "An optional comment")

## End(Not run)
## Not run: experiments.show(cyto_session, 22)

## Not run: experiments.trash(cyto_session, 22)

## Not run: experiments.update(cyto_session, experiment=cyto_experiment)
```

fcs_files *FCS File Endpoints*

Description

Interact with FCS file endpoints.

Usage

```
## S4 method for signature 'UserSession'
fcs_files.download(
  UserSession,
  experiment_id,
  fcs_file_id,
  directory = getwd(),
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
fcs_files.download_zip(
  UserSession,
  experiment_id,
  fcs_files,
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
fcs_files.file_internal_comp_show(
  UserSession,
  experiment_id,
  fcs_file_id,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
fcs_files.list(
  UserSession,
  experiment_id,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
fcs_files.show(
  UserSession,
  experiment_id,
  fcs_file_id,
```

```
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
fcs_files.upload(
  UserSession,
  experiment_id,
  file_path,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
fcs_files.upload_zip(
  UserSession,
  experiment_id,
  file_path,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
fcs_files.status(
  UserSession,
  experiment_id,
  timeout = UserSession@long_timeout
)
```

Arguments

`UserSession`	Cytobank UserSession object
`experiment_id`	integer representing an experiment ID
`fcs_file_id`	integer representing an FCS file ID
`directory`	character representing a specific directory to which the file will be downloaded (optional ending directory slash), if left empty, the default will be the current working directory **[optional]**
`timeout`	integer representing the request timeout time in seconds **[optional]**
`fcs_files`	vector/list of integers representing a list of FCS file IDs
`output`	character representing the output format **[optional]** - *fcs_files.file_internal_comp_show :* `("default", "dataframe", "raw")` - *fcs_files.list, fcs_files.show, fcs_files.upload, fcs_files.upload_zip :* `("default", "raw")` - `dataframe`*: converts the file internal compensation matrix output to a dataframe*
`file_path`	character representing a file path

Details

`fcs_files.download` Download an FCS file from an experiment.

`fcs_files.download_zip` Download all or a select set of FCS files as a zip file from an experiment. The download link of the zip file will be sent to the user's registered email address.

`fcs_files.file_internal_comp_show` Show FCS file internal compensation (aka spillover matrix, spill matrix, spill string) details from an experiment.
- *Optional output parameter, specify one of the following:* `("default", "dataframe", "raw")`

`fcs_files.list` List all FCS files from an experiment. Outputs a dataframe [default] or raw full list with all fields present.
- *Optional output parameter, specify one of the following:* `("default", "raw")`

`fcs_files.show` Show FCS file details from an experiment. - *Optional output parameter, specify one of the following:* `("default", "raw")`

`fcs_files.upload` Upload an FCS file to an experiment. Cytobank User ID has to be attached to the UserSession object. See the help document of authenticate function for details. - *Optional output parameter, specify one of the following:* `("default", "raw")`

`fcs_files.upload_zip` Upload a zip of FCS file(s) to an experiment. - *Optional output parameter, specify one of the following:* `("default", "raw")`

`fcs_files.status` Check status of file(s) in an experiment. Return FALSE and print out an warming message if it fail. Otherwise, return a R dataframe object with file status information.

Examples

```
## Not run: # Authenticate via username/password
cyto_session <- authenticate(site="premium", username="cyril_cytometry", password="cytobank_rocks!")
# Authenticate via auth_token
cyto_session <- authenticate(site="premium", auth_token="my_secret_auth_token")

## End(Not run)
## Not run: # Download an FCS file to the current working directory
fcs_files.download(cyto_session, experiment_id = 22, fcs_file_id = 4)

# Download an FCS file to a new directory
fcs_files.download(cyto_session, 22, experiment_id = 22, fcs_file_id = 4,
directory="/my/new/download/directory/")

## End(Not run)
## Not run: # Download all FCS files as a zip file
fcs_files.download_zip(cyto_session, experiment_id=22)

# Download a select set of FCS files as a zip file
fcs_files.download_zip(cyto_session, experiment_id=22, fcs_files=c(22, 23, 24, 25))

## End(Not run)
## Not run: # List of a file internal compensation, containing a file internal compensation matrix
fcs_files.file_internal_comp_show(cyto_session, 22, fcs_file_id=2)

# Dataframe only of a file internal compensation
fcs_files.file_internal_comp_show(cyto_session, 22, fcs_file_id=2, output="dataframe")
```

```
# Raw list of a file internal compensation
fcs_files.file_internal_comp_show(cyto_session, 22, fcs_file_id=2, output="raw")

## End(Not run)
## Not run: # Dataframe of all FCS files with all fields present
fcs_files.list(cyto_session, 22)

# Raw list of all FCS files with all fields present
fcs_files.list(cyto_session, 22, output="raw")

## End(Not run)
## Not run: fcs_files.show(cyto_session, 22, fcs_file_id=2)

## Not run: fcs_files.upload(cyto_session, 22, file_path="/path/to/my_fcs_file.fcs")

## Not run: fcs_files.upload_zip(cyto_session, 22, file_path="/path/to/my_fcs_files.zip")

## Not run: fcs_files.status(cyto_session, 22)
```

`flowsom` *FlowSOM Endpoints*

Description

Interact with FlowSOM advanced analyses using these endpoints.

Usage

```
## S4 method for signature 'UserSession,FlowSOM'
flowsom.copy_settings(
  UserSession,
  flowsom,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession,FlowSOM'
flowsom.delete(UserSession, flowsom, timeout = UserSession@short_timeout)

## S4 method for signature 'UserSession,FlowSOM'
flowsom.download(
  UserSession,
  flowsom,
  directory = getwd(),
  timeout = UserSession@long_timeout
)
```

```
## S4 method for signature 'UserSession'
flowsom.list(
  UserSession,
  experiment_id,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
flowsom.new(
  UserSession,
  experiment_id,
  flowsom_name,
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession,FlowSOM'
flowsom.rename(
  UserSession,
  flowsom,
  flowsom_name,
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession,FlowSOM'
flowsom.run(
  UserSession,
  flowsom,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
flowsom.show(
  UserSession,
  experiment_id,
  flowsom_id,
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession,FlowSOM'
flowsom.status(
  UserSession,
  flowsom,
  output = "default",
  timeout = UserSession@long_timeout
)
```

```
## S4 method for signature 'UserSession,FlowSOM'
flowsom.update(UserSession, flowsom, timeout = UserSession@long_timeout)
```

Arguments

UserSession	Cytobank UserSession object
flowsom	Cytobank FlowSOM object
output	character representing the output format **[optional]** - *flowsom.list, flowsom.run, flowsom.status :* ("default", "raw")
timeout	integer representing the request timeout time in seconds **[optional]**
directory	character representing a specific directory to which the file will be downloaded (optional ending directory slash), if left empty, the default will be the current working directory **[optional]**
experiment_id	integer representing an experiment ID
flowsom_name	character representing a new FlowSOM name
flowsom_id	integer representing a FlowSOM ID

Details

flowsom.copy_settings Copy FlowSOM advanced analysis settings from an experiment and returns a FlowSOM object.

flowsom.delete Delete a FlowSOM advanced analysis from an experiment.

flowsom.download Download a FlowSOM analysis from an experiment.

flowsom.list List all FlowSOM advanced analyses from an experiment. Outputs a dataframe [default] or list with all fields present.
- *Optional output parameter, specify one of the following:* ("default", "raw")

flowsom.new Create a new FlowSOM advanced analysis from an experiment and returns a FlowSOM object.

flowsom.rename Rename a FlowSOM advanced analysis from an experiment and returns a FlowSOM object.

flowsom.run Run a FlowSOM advanced analysis from an experiment.

flowsom.show Show FlowSOM advanced analysis details from an experiment and returns a FlowSOM object.

flowsom.status Show the status of a FlowSOM advanced analysis from an experiment.

flowsom.update Update a FlowSOM advanced analysis from an experiment and returns the new FlowSOM object.

Examples

```
## Not run: # Authenticate via username/password
cyto_session <- authenticate(site="premium", username="cyril_cytometry", password="cytobank_rocks!")
# Authenticate via auth_token
cyto_session <- authenticate(site="premium", auth_token="my_secret_auth_token")

# cyto_flowsom refers to a FlowSOM object that is created from FlowSOM endpoints
```

```
#   examples: flowsom.new, flowsom.show (see details section for more)

## End(Not run)
## Not run: flowsom.copy_settings(cyto_session, flowsom=cyto_flowsom)

## Not run: flowsom.delete(cyto_session, flowsom=cyto_flowsom)

## Not run: # Download a FlowSOM analysis to the current working directory
flowsom.download(cyto_session, flowsom)

# Download a FlowSOM analysis to a new directory
flowsom.download(cyto_session, flowsom, directory="/my/new/download/directory/")

## End(Not run)
## Not run: # Dataframe of all FlowSOM advanced analyses with all fields present
flowsom.list(cyto_session, 22)

# Raw list of all FlowSOM advanced analyses with all fields present
flowsom.list(cyto_session, 22, output="raw")

## End(Not run)
## Not run: flowsom.new(cyto_session, 22, flowsom_name="My new FlowSOM analysis")

## Not run: flowsom.rename(cyto_session, flowsom=cyto_flowsom,
    flowsom_name="My updated FlowSOM name")

## End(Not run)
## Not run: flowsom.run(cyto_session, flowsom=cyto_flowsom)

## Not run: flowsom.show(cyto_session, 22, flowsom_id=2)

## Not run: flowsom.status(cyto_session, flowsom=cyto_flowsom)

## Not run: flowsom.update(cyto_session, flowsom=cyto_flowsom)
```

`FlowSOM-class` *S4 FlowSOM Class*

Description

A FlowSOM object that holds pertinent FlowSOM advanced analysis run information, learn more about FlowSOM. This class should never be called explicitly. If a user would like to create a new Cytobank FlowSOM object, utilize the flowsom.new function, or any other FlowSOM endpoints that return FlowSOM objects documented in the 'Details' section.

Value

A FlowSOM advanced analysis object

Slots

`attachment_id` numeric representing the FlowSOM attachment to the source experiment containing the FlowSOM results

`author` character representing the author of the FlowSOM analysis

`auto_seed` logical representing whether to set an auto seed value or not

`canceled` logical representing whether or not the FlowSOM analysis is canceled

`channels_to_plot` list representing short channel IDs corresponding to channels to output channel-colored MST plots, learn more about FlowSOM PDF output

`clustering_method` character representing the clustering method
- *choose from the following :* `("consensus" [default], "hierarchical", "kmeans")`

`cluster_size_type` character representing the cluster size type, learn more about FlowSOM PDF output
- *choose from the following :* `("both", "fixed", "relative" [default])`

`completed` logical representing whether or not the FlowSOM analysis is complete

`created_experiment` numeric representing the experiment that gets created from the FlowSOM analysis

`desired_events_per_file` numeric representing the number of desired events per file if `event_sampling_method` is set to `equal`, learn more about FlowSOM event sampling methods

`desired_total_events` numeric representing the total desired number of events to sample amongst all selected files if `event_sampling_method` is set to `proportional`, learn more about FlowSOM event sampling methods

`event_sampling_method` character representing the FlowSOM sampling method, learn more about FlowSOM event sampling methods
- *choose from the following :* `("all", "equal" [default], "proportional")`

`expected_clusters` numeric representing the number of expected clusters, learn more about choosing target number of clusters for FlowSOM

`expected_metaclusters` numeric representing the expected number of metaclusters learn more about choosing target number of metaclusters for FlowSOM

`external_som_analysis_info` character representing FlowSOM analysis information

`external_som_analysis_id` character representing the ID of a corresponding FlowSOM analysis ID if `som_creation_method` set to `"import_existing"`

`external_som_attachment_id` character representing the ID of a corresponding completed FlowSOM analysis if `som_creation_method` is set to `import_existing`

`fcs_files` list of integers or character representing a list of FCS file IDs

`final_result` character representing whether or not the FlowSOM analysis is successful

`fixed_cluster_size` integer representing fixed cluster size if `cluster_size_type` set to `"fixed"` or `"both"`learn more about FlowSOM PDF output

`flowsom_id` numeric representing the FlowSOM analysis ID

`gate_set_names_to_label` list of character representing populations to label in the population pie plots, learn more about FlowSOM PDF output

`iterations` numeric representing the number of times FlowSOM processes the dataset using its step-wise optimization algorithm, learn more about iterations in FlowSOM

`max_relative_cluster_size` numeric representing the max relative cluster size (only applicable if `cluster_size_type` set to `"relative"` or `"both"`, learn more about FlowSOM PDF output

`normalize_scales` logical representing whether or not to normalize scales

`num_events_to_actually_sample` numeric representing the events actually sampled

`num_fcs_files` numeric representing the number of FCS files

`output_file_type` character representing the output file type
- *choose from the following :* `("both", "pdf" [default], "png")`

`population_id` integer representing a population **gate set ID**

`random_seed` numeric representing the seed value learn more about setting the seed for FlowSOM

`show_background_on_legend` logical representing whether or not to show background on legend, learn more about FlowSOM PDF output

`show_background_on_channel_colored_msts` logical representing whether or not to show background on channel colored MSTs, learn more about FlowSOM PDF output

`show_background_on_population_pies` logical representing whether or not to show background on population pies, learn more about FlowSOM PDF output

`som_creation_method` character representing the FlowSOM creation method, learn more about SOM creationg methods for FlowSOM
- *choose from the following :* `("create_new" [default], "import_existing")`

`type` character

`gates` *Gate Endpoints*

Description

Interact with gate endpoints. In Cytobank there is a distinction between gates and populations. A gate is simply a shape drawn on a plot. A population is a set of gates and can have parents and children. Learn more about gates and populations. Currently, gate and population information can only be read and not written to Cytobank via the JSON API. To write gates and populations to Cytobank via the API, the gates.gatingML_upload endpoint should be used.

Usage

```
## S4 method for signature 'UserSession'
gates.gatingML_download(
  UserSession,
  experiment_id,
  directory = getwd(),
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
gates.gatingML_upload(
```

```
  UserSession,
  experiment_id,
  file_path,
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
gates.apply(UserSession, experiment_id, timeout = UserSession@long_timeout)

## S4 method for signature 'UserSession'
gates.list(
  UserSession,
  experiment_id,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
gates.show(
  UserSession,
  experiment_id,
  gate_id,
  output = "default",
  timeout = UserSession@short_timeout
)
```

Arguments

`UserSession`	Cytobank UserSession object
`experiment_id`	integer representing an experiment ID
`directory`	character representing a specific directory to which the file will be downloaded (optional ending directory slash), if left empty, the default will be the current working directory **[optional]**
`timeout`	integer representing the request timeout time in seconds **[optional]**
`file_path`	character representing a file path
`output`	character representing the output format **[optional]** - *gates.list, gates.show :* `("default", "raw")`
`gate_id`	integer representing a gate ID

Details

`gates.gatingML_download` Download the gatingML from an experiment. Learn more about Gating-ML.

`gates.gatingML_upload` Upload a gatingML to an experiment. Learn more about Gating-ML.

`gates.apply` Apply gates as Experiment Gates. Gates must be applied in order for Scratch Gates to be converted to Experiment gates. Experiment gates are used for generating statistics, illustrations, and advanced analyses. Learn more about applying gates.

`gates.list` List all gates from an experiment. Outputs a dataframe [default] or raw list with all fields present. Currently only the Scratch Gates from the gating interface are returned. These have a version of -1. This is to be contrasted with Experiment Gates, which will have a version number that is a positive integer equal to the number of times the version has been incremented in the gating interface. Learn more about gate versioning in Cytobank.
- *Optional output parameter, specify one of the following:* `("default", "raw")`

`gates.show` Show gate details from an experiment.

Examples

```
## Not run: # Authenticate via username/password
cyto_session <- authenticate(site="premium", username="cyril_cytometry", password="cytobank_rocks!")
# Authenticate via auth_token
cyto_session <- authenticate(site="premium", auth_token="my_secret_auth_token")

## End(Not run)
## Not run: gates.gatingML_download(cyto_session, 22, directory="/my/new/download/directory/")

## Not run: gates.gatingML_upload(cyto_session, 22, file_path="/path/to/my_gatingML.xml")

## Not run: gates.apply(cyto_session, 22)

## Not run: # Dataframe of all gates with all fields present
gates.list(cyto_session, 22)

# Raw list of all gates with all fields present
gates.list(cyto_session, 22, output="raw")

## End(Not run)
## Not run: gates.show(cyto_session, 22, gate_id=2)
```

`helper_functions` *Helper Functions*

Description

Various helper functions to utilize within the Cytobank API.

Usage

```
helper.filter_names_to_ids_from_df(ids_names_df, names_array = c("*"))

helper.channel_ids_from_long_names(
  panels_list,
  long_channel_names,
  fcs_files = c()
)
```

Arguments

ids_names_df	dataframe containing both IDs and their associated names
names_array	vector or list of character regular expressions to use
panels_list	list provided from the panels.list endpoint
long_channel_names	vector of character representing long channel names
fcs_files	vector of integers representing a list of FCS file IDs

Details

helper.filter_names_to_ids_from_df Compile a vector of IDs from an array of regular expressions.

helper.channel_ids_from_long_names Compile a vector of IDs based on long channel names for specific FCS files from an experiment. If no FCS files are provided, IDs will be retrieved based on unique short channel / long channel combinations across all FCS files.

Examples

```
## Not run: helper.filter_names_to_ids_from_df(id_and_names_dataframe,
names_list=c("CD.*", "Time", "pp38"))

## End(Not run)
## Not run: helper.channel_ids_from_long_names(panels.list(cyto_session, 22),
  long_channel_names=c("long_channel1", "long_channel2"), fcs_files=c(1,2,3,4,5))

## End(Not run)
```

---

news *News*

Description

Get news on CytobankAPI updates

Usage

```
CytobankAPI_news()
```

Details

CytobankAPI_news View a log of CytobankAPI updates and release notes.

`optSNE-class` *S4 opt-SNE Class*

Description

A opt-SNE object that holds pertinent opt-SNE advanced analysis run information. This class should never be called explicitly. If a user would like to create a new Cytobank opt-SNE object, utilize the dimensionality_reduction.new function, or any other opt-SNE endpoints that return opt-SNE objects documented in the 'Details' section.

Value

A Dimensionality Reduction advanced analysis object

Slots

`perplexity` numeric representing a rough guess for the number of close neighbors any given cellular event will have, learn more about Dimensionality Reduction perplexity

`auto_learning_rate` logical representing whether or not to set auto learning rate

`clustering_channels` list the channels selected for the Dimensionality Reduction analysis, this can be either a list of short channel IDs (integer) OR long channel names (character)

`desired_events_per_file` numeric representing the number of desired events per file

`desired_total_events` numeric representing the number of desired total events per file

`early_exaggeration` numeric representing how tight natural clusters in the original space are in the embedded space and how much space will be between them

`event_sampling_method` character representing the name of event sampling method will be used, learn more about Event Sampling for Dimensionality Reduction analysis

`fcsfile_ids` list representing the fcs file ids

`gateset_id` numeric representing the selected gate id

`learning_rate` numeric representing the learning rate,learn more about opt-SNE learning rate.

`max_iterations` numeric representing the maximum number of iterations to perform– typically opt-SNE will automatically stop before this number is reached

`normalize_scales` logical representing whether or not to normalize scales

`random_seed` numeric representing the seed, Dimensionality Reduction picks a random seed each run, but if users want reproducible data, setting the same seed will allow them to do this

`panels` *Panel Endpoints*

Description

Interact with panel endpoints. A collection of channels, the markers being studied on them, and the FCS files this applies to form a panel. Learn more about panels in Cytobank.

Usage

```
## S4 method for signature 'UserSession'
panels.list(
  UserSession,
  experiment_id,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
panels.show(
  UserSession,
  experiment_id,
  panel_id,
  output = "default",
  timeout = UserSession@short_timeout
)
```

Arguments

`UserSession`	Cytobank UserSession object
`experiment_id`	integer representing an experiment ID
`output`	character representing the output format **[optional]** - *panels.list, panels.show :* `("default", "raw")`
`timeout`	integer representing the request timeout time in seconds **[optional]**
`panel_id`	integer representing a panel ID

Details

`panels.list` List all panels from an experiment. Outputs a formatted list [default] or raw list with all fields present.
- *Optional output parameter, specify one of the following:* `("default", "raw")`

`panels.show` Show panel details from an experiment. Outputs a full list with all fields present, or an IDs/names list (See attachments examples section for IDs/names list example).
- *Optional output parameter, specify one of the following:* `("default", "raw")`

Examples

```
## Not run: # Authenticate via username/password
cyto_session <- authenticate(site="premium", username="cyril_cytometry", password="cytobank_rocks!")
# Authenticate via auth_token
cyto_session <- authenticate(site="premium", auth_token="my_secret_auth_token")

## End(Not run)
## Not run: # Full panel list with all fields present, with a dataframe of channels
panels.list(cyto_session, 22)

# Raw list of all panels with all fields present
panels.list(cyto_session, 22, output="raw")

## End(Not run)
## Not run: # Full panel info with all fields present
panels.show(cyto_session, 22, panel_id=2)

## End(Not run)
```

---

peacoqc *PeacoQC Endpoints*

Description

Interact with PeacoQC using these endpoints.

Usage

```
## S4 method for signature 'UserSession,PeacoQC'
peacoqc.copy_settings(
  UserSession,
  peacoqc,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
peacoqc.list(
  UserSession,
  experiment_id,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
peacoqc.new(
  UserSession,
  experiment_id,
```

```
  peaco_qc_name,
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession,PeacoQC'
peacoqc.rename(
  UserSession,
  peacoqc,
  peaco_qc_name,
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession,PeacoQC'
peacoqc.run(
  UserSession,
  peacoqc,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
peacoqc.show(
  UserSession,
  experiment_id,
  peaco_qc_id,
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession,PeacoQC'
peacoqc.status(
  UserSession,
  peacoqc,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession,PeacoQC'
peacoqc.update(UserSession, peacoqc, timeout = UserSession@long_timeout)
```

Arguments

`UserSession`	Cytobank UserSession object
`peacoqc`	Cytobank PeacoQC object
`output`	character representing the output format **[optional]** - *peacoqc.list, peacoqc.run, peacoqc.status, peacoqc.copy_settings :* `("default", "raw")`
`timeout`	integer representing the request timeout time in seconds **[optional]**
`experiment_id`	integer representing an experiment ID

peaco_qc_name	character representing a new PeacoQC name
peaco_qc_id	integer representing a PeacoQC ID

Details

peacoqc.copy_settings Copy PeacoQC settings from an experiment and returns a PeacoQC object.

peacoqc.list List all PeacoQC from an experiment. Outputs a dataframe [default] or list with all fields present.
- *Optional output parameter, specify one of the following:* ("default", "raw")

peacoqc.new Create a new PeacoQC advanced analysis from an experiment and returns a PeacoQC object.

peacoqc.rename Rename a PeacoQC from an experiment and returns a PeacoQC object.

peacoqc.run Run a PeacoQC from an experiment.

peacoqc.show Show PeacoQC details from an experiment and returns a PeacoQC object.

peacoqc.status Show the status of a PeacoQC from an experiment.

peacoqc.update Update a PeacoQC from an experiment and returns the new PeacoQC object.

Examples

```
## Not run: # Authenticate via username/password
cyto_session <- authenticate(site="premium", username="cyril_cytometry", password="cytobank_rocks!")
# Authenticate via auth_token
cyto_session <- authenticate(site="premium", auth_token="my_secret_auth_token")

# cyto_peacoqc refers to a PeacoQC object that is created from PeacoQC endpoints
#   examples: peacoqc.new, peacoqc.show (see details section for more)

## End(Not run)
## Not run: peacoqc.copy_settings(cyto_session, peacoqc=cyto_peacoqc)

## Not run: # Dataframe of all PeacoQCs with all fields present
peacoqc.list(cyto_session, 22)

# Raw list of all PeacoQCs with all fields present
peacoqc.list(cyto_session, 22, output="raw")

## End(Not run)
## Not run: peacoqc.new(cyto_session, 22, peaco_qc_name="My new PeacoQC")

## Not run: peacoqc.rename(cyto_session, peacoqc=cyto_peacoqc,
    peaco_qc_name="My updated PeacoQC name")

## End(Not run)
## Not run: peacoqc.run(cyto_session, peacoqc=cyto_peacoqc)

## Not run: peacoqc.show(cyto_session, experiment_id=22, peaco_qc_id=2)

## Not run: peacoqc.status(cyto_session, peacoqc=cyto_peacoqc)
```

```
## Not run: peacoqc.update(cyto_session, peacoqc=cyto_peacoqc)
```

`PeacoQC-class` *S4 PeacoQC Class*

Description

A PeacoQC object that holds pertinent PeacoQC data QC run information This class should never be called explicitly. If a user would like to create a new Cytobank PeacoQC object, utilize the peacoqc.new function, or any other PeacoQC endpoints that return PeacoQC objects documented in the 'Details' section.

Value

A PeacoQC object

Slots

`author` character representing the author of the PeacoQC analysis

`attachment_id` numeric representing the PeacoQC attachment to the source experiment containing the PeacoQC results

`channel_unique_identifiers` list of character representing a list of unique channel identifiers

`compensation_id` the compensation ID selected for the PeacoQC data QC

`completed` logical representing whether or not the PeacoQC is complete

`consecutive_bins` numeric if 'good' bins are located between bins that are removed, they will also be marked as 'bad'. Can be set to any integer between 1 and 50 (inclusive)

`detection_method` character representing the method(s) used to detect and filter out anomalies. - *choose from the following :* `("all" [default], "IT", "MAD")`

`errors` list of character representing a list of error messages of the PeacoQC

`failed` logical representing whether or not the PeacoQC is failed

`fcs_files` list of integers or character representing a list of FCS file IDs

`final_result` character representing whether or not the PeacoQC is successful

`heatmap_attachment_id` numeric representing the PeacoQC heatmap image attachment to the source experiment

`it_limit` numeric representing the IsolationTree parameter. Higher values mean the IT method will be less strict. Can be set to any float between 0.2 and 1.0(inclusive)

`mad` numeric representing the MAD parameter. Higher values mean the MAD method will be less strict. Can be set to any integer between 1 and 100 (inclusive)

`max_bins` numeric representing the maximum number of bins that can be used in the cleaning process. If this value is lowered, larger bins will be made. Can be set to any integer between 40 and 1,000,000 (inclusive)

`name` the name of the advanced analysis

`peaco_qc_id` numeric representing the PeacoQC ID

`remove_margins` if the value is true, they will remove margin events based on the internal description of the fcs file. Can be set to a boolean value

`source_experiment` the source experiment ID the advanced analysis is associated with

`status` character representing the status of the advanced analysis

`type` character

`use_internal_scales_for_margins` logical this parameter is required when removeMargins is set to true. Set to true, the events will transtorm with fcs file internal scales. Set to false, the events will transform with cytobank scales. Can be set to a boolean value

`validFcsFileIds` list of integers or character representing a list of valid FCS file IDs can run PeacoQC

`populations` *Population Endpoints*

Description

Interact with population (aka gate sets) endpoints. A population is a set of gates and can have parents and children. Learn more about gates and populations.

Usage

```
## S4 method for signature 'UserSession'
populations.list(
  UserSession,
  experiment_id,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
populations.show(
  UserSession,
  experiment_id,
  population_id,
  output = "default",
  timeout = UserSession@short_timeout
)
```

Arguments

`UserSession`	Cytobank UserSession object
`experiment_id`	integer representing an experiment ID

output	character representing the output format **[optional]** - *populations.list, populations.show :* ("default", "raw")
timeout	integer representing the request timeout time in seconds
population_id	integer representing a population ID

Details

populations.list List all populations from an experiment. Outputs a dataframe [default] or raw list with all fields present.
- *Optional output parameter, specify one of the following:* ("default", "raw")

populations.show Show population details from an experiment. - *Optional output parameter, specify one of the following:* ("default", "raw")

Examples

```
## Not run: # Authenticate via username/password
cyto_session <- authenticate(site="premium", username="cyril_cytometry", password="cytobank_rocks!")
# Authenticate via auth_token
cyto_session <- authenticate(site="premium", auth_token="my_secret_auth_token")

## End(Not run)
## Not run: # Dataframe of all populations with all fields present
populations.list(cyto_session, 22)

# Raw list of all populations with all fields present
populations.list(cyto_session, 22, output="raw")

## End(Not run)
## Not run: populations.show(cyto_session, 22, population_id=2)
```

---

sample_tags *Sample Tag Endpoints*

Description

Interact with sample tag endpoints. Download and upload sample tags to save time during the annotation process. Learn more about sample tags here.

Usage

```
## S4 method for signature 'UserSession'
sample_tags.download(
  UserSession,
  experiment_id,
  directory = getwd(),
  timeout = UserSession@short_timeout
)
```

```
## S4 method for signature 'UserSession'
sample_tags.upload(
  UserSession,
  experiment_id,
  file_path,
  timeout = UserSession@long_timeout
)
```

Arguments

UserSession	Cytobank UserSession object
experiment_id	integer representing an experiment ID
directory	character representing a specific directory to which the file will be downloaded (optional ending directory slash), if left empty, the default will be the current working directory **[optional]**
timeout	integer representing the request timeout time in seconds
file_path	character representing a file path

Details

`sample_tags.download` Download the sample tags from an experiment.

`sample_tags.upload` Upload sample tag annotation data TSV to an experiment.

Examples

```
## Not run: # Authenticate via username/password
cyto_session <- authenticate(site="premium", username="cyril_cytometry", password="cytobank_rocks!")
# Authenticate via auth_token
cyto_session <- authenticate(site="premium", auth_token="my_secret_auth_token")

## End(Not run)
## Not run: # Download the experiment sample tags TSV to the current working directory
sample_tags.download(cyto_session, 22)

# Download the experiment sample tags TSV to a new directory
sample_tags.download(cyto_session, 22, directory="/my/new/download/directory/")

## End(Not run)
## Not run: sample_tags.upload(cyto_session, 22, file_path="/path/to/my_annotations.tsv")
```

scales *Scale Endpoints*

Description

Interact with scale endpoints. Data are rarely presented exactly as they were acquired on the instrument. Learn more about data scaling.

Usage

```
## S4 method for signature 'UserSession'
scales.list(
  UserSession,
  experiment_id,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
scales.show(
  UserSession,
  experiment_id,
  scale_id,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
scales.update(
  UserSession,
  scale,
  output = "default",
  timeout = UserSession@short_timeout
)
```

Arguments

UserSession	Cytobank UserSession object
experiment_id	integer representing an experiment ID
output	character representing the output format **[optional]** - *scales.list, scales.show, scales.update :* ("default", "raw")
timeout	integer representing the request timeout time in seconds
scale_id	integer representing a scale ID
scale	dataframe representing a scale

Details

`scales.list` List all scales from an experiment. Outputs a dataframe [default] or raw list with all fields present.
- *Optional output parameter, specify one of the following:* `("default", "raw")`

`scales.show` Show scale details from an experiment. - *Optional output parameter, specify one of the following:* `("default", "raw")`

`scales.update` Update a single scale from an experiment. (all parameters are optional, except for experiment_id and scale_id)
- *Scale Types* – `1: Linear, 2: Log, 4: Arcsinh`
- *Optional output parameter, specify one of the following:* `("default", "raw")`

Examples

```
## Not run: # Authenticate via username/password
cyto_session <- authenticate(site="premium", username="cyril_cytometry", password="cytobank_rocks!")
# Authenticate via auth_token
cyto_session <- authenticate(site="premium", auth_token="my_secret_auth_token")

## End(Not run)
## Not run: # Dataframe of all scales with all fields present
scales.list(cyto_session, 22)

# Raw list of all scales with all fields present
scales.list(cyto_session, 22, output="raw")

## End(Not run)
## Not run: scales.show(cyto_session, 22, scale_id=2)

## Not run: # Update any number of parameters (scale_type, cofactor, minimum, maximum)
# Scale Types -- 1: Linear, 2: Log, 4: Arcsinh
scales.update(cyto_session, scale=cyto_scale)

## End(Not run)
```

---

spade *SPADE Endpoints*

Description

Interact with SPADE advanced analyses using these endpoints.

Usage

```
## S4 method for signature 'UserSession,SPADE'
spade.bubbles_export(
  UserSession,
  spade,
  bubbles,
```

```
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession,SPADE'
spade.bubbles_set(
  UserSession,
  spade,
  bubbles,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession,SPADE'
spade.bubbles_show(
  UserSession,
  spade,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession,SPADE'
spade.copy_results(
  UserSession,
  spade,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession,SPADE'
spade.copy_settings(
  UserSession,
  spade,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession,SPADE'
spade.delete(UserSession, spade, timeout = UserSession@short_timeout)

## S4 method for signature 'UserSession,SPADE'
spade.download_all(
  UserSession,
  spade,
  directory = getwd(),
  timeout = UserSession@long_timeout
)
```

```
## S4 method for signature 'UserSession,SPADE'
spade.download_clusters_table(
  UserSession,
  spade,
  directory = getwd(),
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession,SPADE'
spade.download_global_boundaries_table(
  UserSession,
  spade,
  directory = getwd(),
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession,SPADE'
spade.download_gml(
  UserSession,
  spade,
  directory = getwd(),
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession,SPADE'
spade.download_layout_table(
  UserSession,
  spade,
  directory = getwd(),
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession,SPADE'
spade.download_statistics_tables(
  UserSession,
  spade,
  directory = getwd(),
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
spade.list(
  UserSession,
  experiment_id,
  output = "default",
  timeout = UserSession@short_timeout
)
```

```
## S4 method for signature 'UserSession'
spade.new(
  UserSession,
  experiment_id,
  spade_name,
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession,SPADE'
spade.rename(
  UserSession,
  spade,
  spade_name,
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession,SPADE'
spade.run(
  UserSession,
  spade,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
spade.show(
  UserSession,
  experiment_id,
  spade_id,
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession,SPADE'
spade.status(
  UserSession,
  spade,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession,SPADE'
spade.update(UserSession, spade, timeout = UserSession@long_timeout)
```

Arguments

UserSession	Cytobank UserSession object
spade	Cytobank SPADE object
bubbles	vector/list of characters representing bubbles within a SPADE analysis, learn

	more about SPADE bubbles
`output`	character representing the output format **[optional]** - *spade.list, spade.run, spade.status :* `("default", "raw")`
`timeout`	integer representing the request timeout time in seconds **[optional]**
`directory`	character representing a specific directory (optional ending directory slash), default will be current working directory **[optional]**
`experiment_id`	integer representing an experiment ID
`spade_name`	character representing a new SPADE name
`spade_id`	integer representing a SPADE ID

Details

`spade.bubbles_export` Export SPADE advanced analysis bubbles from an experiment to a new experiment.

`spade.bubbles_set` Set SPADE advanced analysis bubbles from an experiment.

`spade.bubbles_show` Show SPADE advanced analysis bubbles from an experiment.

`spade.copy_results` Copy SPADE advanced analysis results from an experiment to a new experiment.

`spade.copy_settings` Copy SPADE advanced analysis settings from an experiment.

`spade.delete` Delete a SPADE advanced analysis from an experiment.

`spade.download_all` Download a SPADE advanced analysis with all data included from an experiment.

`spade.download_clusters_table` Download a SPADE advanced analysis global clusters table from an experiment.

`spade.download_global_boundaries_table` Download a SPADE advanced analysis global boundaries table from an experiment.

`spade.download_gml` Download a SPADE advanced analysis GML from an experiment.

`spade.download_layout_table` Download a SPADE advanced analysis layout table from an experiment.

`spade.download_statistics_tables` Download a SPADE advanced analysis statistics table from an experiment.

`spade.list` List all SPADE advanced analyses from an experiment. Outputs a dataframe [default] or list with all fields present.
- *Optional output parameter, specify one of the following:* `("default", "raw")`

`spade.new` Create a new SPADE advanced analysis from an experiment and returns a SPADE object.

`spade.rename` Rename a SPADE advanced analysis from an experiment and returns a SPADE object.

`spade.run` Run a SPADE advanced analysis from an experiment.

`spade.show` Show SPADE advanced analysis details from an experiment and returns a SPADE object.

spade.status Show the status of a SPADE advanced analysis from an experiment.

spade.update Update a SPADE advanced analysis from an experiment and returns the new SPADE object.

Examples

```
## Not run: # Authenticate via username/password
cyto_session <- authenticate(site="premium", username="cyril_cytometry", password="cytobank_rocks!")
# Authenticate via auth_token
cyto_session <- authenticate(site="premium", auth_token="my_secret_auth_token")

# cyto_spade refers to a SPADE object that is created from SPADE endpoints
#   examples: spade.new, spade.show (see details section for more)

## End(Not run)
## Not run: spade.bubbles_export(cyto_session, spade=cyto_spade, bubbles=c("bubble1", "bubble2"))

## Not run: named_bubble_list_of_node_vectors <- list("bubble_1"=c(1,2,4), "bubble_2"=8, "bubble_4"=c(10,12))
spade.bubbles_set(cyto_session, spade=cyto_spade, bubbles=named_bubble_list_of_node_vectors)

## End(Not run)
## Not run: spade.bubbles_show(cyto_session, spade=cyto_spade)

## Not run: spade.copy_results(cyto_session, spade=cyto_spade)

## Not run: spade.copy_settings(cyto_session, spade=cyto_spade)

## Not run: spade.delete(cyto_session, spade=cyto_spade)

## Not run: spade.download_all(cyto_session, spade=cyto_spade,
    directory="/my/new/download/directory/")

## End(Not run)
## Not run: spade.download_clusters_table(cyto_session, spade=cyto_spade,
  directory="/my/new/download/directory/")

## End(Not run)
## Not run: spade.download_global_boundaries_table(cyto_session,
  spade=cyto_spade, directory="/my/new/download/directory/")

## End(Not run)
## Not run: spade.download_gml(cyto_session, spade=cyto_spade,
    directory="/my/new/download/directory/")

## End(Not run)
## Not run: spade.download_layout_table(cyto_session, spade=cyto_spade,
    directory="/my/new/download/directory/")

## End(Not run)
## Not run: spade.download_statistics_tables(cyto_session, spade=cyto_spade,
  directory="/my/new/download/directory/")
```

```
## End(Not run)
## Not run: # Dataframe of all SPADE advanced analyses with all fields present
spade.list(cyto_session, 22)

# Raw list of all SPADE advanced analyses with all fields present
spade.list(cyto_session, 22, output="raw")

## End(Not run)
## Not run: spade.new(cyto_session, 22, spade_name="My new SPADE analysis")

## Not run: spade.rename(cyto_session, spade=cyto_spade, spade_name="My updated SPADE name")

## Not run: spade.run(cyto_session, spade=cyto_spade)

## Not run: spade.show(cyto_session, 22, spade_id=2)

## Not run: spade.status(cyto_session, spade=cyto_spade)

## Not run: spade.update(cyto_session, spade=cyto_spade)
```

`SPADE-class` *S4 SPADE Class*

Description

A SPADE object that holds pertinent SPADE advanced analysis run information. This class should never be called explicitly. If a user would like to create a new Cytobank SPADE object, utilize the spade.new function, or any other SPADE endpoints that return SPADE objects documented in the 'Details' section.

Value

A SPADE advanced analysis object

Slots

`created_experiment` numeric representing the experiment that gets created from the SPADE analysis

`down_sampled_events_target` numeric representing the percent OR absolute number (depends on 'down_sampled_events_type' slot) for downsampling occurring within the SPADE analysis, learn more about SPADE density-dependent downsampling

`down_sampled_events_type` character representing the downsampling type for down_sampled_events_target, learn more about SPADE density-dependent downsampling types - *choose one of the following :* (`"percent"` `[default]`, `"absolute_number"`)

`fold_change_groups` dataframe representing the fold change groups within a SPADE analysis, learn more about SPADE fold change groups

`population_id` numeric representing the population to run the SPADE analysis on, learn more about choosing a population for SPADE

`spade_id` numeric representing the SPADE analysis ID

`target_number_nodes` numeric representing how many population nodes SPADE will seek out within the given data, learn more about target number of nodes for SPADE

`statistics` *Statistic Endpoints*

Description

Interact with statistic endpoints. Gather data about event counts and general channel statistics. Create dataframes of statistics to help with visualization and downstream analysis.

Usage

```
## S4 method for signature 'UserSession'
statistics.event_counts(
  UserSession,
  experiment_id,
  gate_version = -1,
  compensation_id,
  fcs_files,
  populations = c(),
  output = "dataframe",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
statistics.general(
  UserSession,
  experiment_id,
  gate_version = -1,
  compensation_id,
  fcs_files,
  channels,
  populations = c(),
  output = "dataframe_row",
  timeout = UserSession@long_timeout
)
```

Arguments

`UserSession`	Cytobank UserSession object
`experiment_id`	integer representing an experiment ID

`gate_version` integer representing an experiment gate version, an integer of -1 corresponds to the state of gates and populations in the gating interface. Faster performance can be achieved by using the maximum gate version from the experiment (learn more about gate versions). Maximum gate version can be seen as the **gateVersion** attribute returned from a call to the Show Experiment Details endpoint **[optional]**

`compensation_id` integer representing a compensation ID (use -2 for file-internal compensation, -1 for uncompensated)

`fcs_files` vector/list of integers representing a list of FCS file IDs

`populations` vector/list of integers representing a list of population IDs to calculate statistics for. This is the **gateSetId** attribute of a population object. Another term for a population is a "gate set". If not specified, all population statistics will be fetched **[optional]**

`output` character representing the output format **[optional]**
- *statistics.event_counts:* `("default" [default], "dataframe")`
- *statistics.general:* `("default", "dataframe_col", "dataframe_row")`
- `dataframe`*: converts the output to a dataframe for the event count statistics*
- `dataframe_col`*: for statistics data on multiple channels, proliferate channel statistics as columns*
- `dataframe_row`*: for statistics data on multiple channels, proliferate channel statistics as rows*

`timeout` integer representing the request timeout time in seconds

`channels` vector/list of integers or character representing a list of channel IDs (integers) or long channel names (character)

Details

`statistics.event_counts` Get event count statistics from an experiment. In the absence of channel information, only event count data are returned. If only event count data are needed, this approach can be faster than retrieving all statistics by avoiding unnecessary computation.
- *Optional output parameter, specify one of the following:* `("full", "dataframe" [default])`
- `dataframe`*: converts the output to a dataframe for the event count statistics*

`statistics.general` Get a batch of common statistics for specific channels on populations from an experiment.
- *Optional output parameter, specify one of the following:* `("full", "dataframe_col", "dataframe_row" [default])`
- `dataframe_col`*: for statistics data on multiple channels, proliferate channel statistics as columns*
- `dataframe_row`*: for statistics data on multiple channels, proliferate channel statistics as rows*

Examples

```
## Not run: # Authenticate via username/password
cyto_session <- authenticate(site="premium", username="cyril_cytometry", password="cytobank_rocks!")
# Authenticate via auth_token
cyto_session <- authenticate(site="premium", auth_token="my_secret_auth_token")
```

```
## End(Not run)
## Not run: statistics.event_counts(cyto_session, 22, compensation_id=-2,
  fcs_files=c(12, 13, 14), channels=c(53, 54, 55), populations=c(32, 33, 34))

## End(Not run)
## Not run: # Full list with all fields present
statistics.general(cyto_session, 22, compensation_id=-2,
  fcs_files=c(12, 13, 14), channels=c(53, 54, 55), populations=c(32, 33, 34))

# Statistics list transformed into a dataframe, proliferating channel statistics by column
statistics.general(cyto_session, 22, compensation_id=-2,
 fcs_files=c(12, 13, 14), channels=c(53, 54, 55), populations=c(32, 33), output="dataframe_col")

# Statistics list transformed into a dataframe, proliferating channel statistics by row
statistics.general(cyto_session, 22, compensation_id=-2,
 fcs_files=c(12, 13, 14), channels=c(53, 54, 55), populations=c(32, 33), output="dataframe_row")

# Statistics list transformed into a dataframe, using helper functions (names_to_ids)
# Get FCS files that match 'pbmc' in their filename
fcs_files <- fcs_files.list(cyto_session, 22)
fcs_files <- fcs_files[,c("id", "filename")]
fcs_files <- unlist(fcs_files$id[grep("pbmc", fcs_files$filename)])

# Get channels that match 'pp' or 'pStat' as their longName
channels <- panels.list(cyto_session, 22)$`Panel 1`$channels
channels <- channels[,c("normalizedShortNameId", "shortName", "longName")]
channels <- channels$normalizedShortNameId[grep("pp.*|pStat.*", channels$longName)]

# Get populations that match 'CD' as their population name
populations <- populations.list(cyto_session, 22)
populations <- populations[,c("gateSetId", "name")]
populations <- populations$id[grep("CD.*", populations$name)]

statistics.general(cyto_session, 22, compensation_id=-2,
 fcs_files=fcs_files, channels=channels, populations=populations, output="dataframe_row")

## End(Not run)
```

---

tSNE-class *S4 tSNE Class*

Description

A tSNE object that holds pertinent tSNE advanced analysis run information. This class should never be called explicitly. If a user would like to create a new Cytobank Dimensionality Reduction object, utilize the dimensionality_reduction.new function, or any other Dimensionality Reduction endpoints that return Dimensionality Reduction objects documented in the 'Details' section.

Value

A Dimensionality Reduction advanced analysis object

Slots

`iterations` numeric representing the number of times Dimensionality Reduction processes the dataset using its step-wise optimization algorithm, learn more about how iterations affect Dimensionality Reduction results

`perplexity` numeric representing a rough guess for the number of close neighbors any given cellular event will have, learn more about Dimensionality Reduction perplexity

`auto_iterations` logical representing whether or not to set auto interations

`auto_learning_rate` logical representing whether or not to set auto learning rate

`clustering_channels` list the channels selected for the Dimensionality Reduction analysis, this can be either a list of short channel IDs (integer) OR long channel names (character)

`desired_events_per_file` numeric representing the number of desired events per file

`desired_total_events` numeric representing the number of desired total events per file

`early_exaggeration` numeric representing how tight natural clusters in the original space are in the embedded space and how much space will be between them

`event_sampling_method` character representing the name of event sampling method will be used, learn more about Event Sampling for Dimensionality Reduction analysis

`fcsfile_ids` list representing the fcs file ids

`gateset_id` numeric representing the selected gate id

`learning_rate` numeric representing the learning rate

`normalize_scales` logical representing whether or not to normalize scales

`UMAP-class` *S4 UMAP Class*

Description

A UMAP object that holds pertinent UMAP advanced analysis run information. This class should never be called explicitly. If a user would like to create a new Cytobank UMAP object, utilize the dimensionality_reduction.new function, or any other UMAP endpoints that return UMAP objects documented in the 'Details' section.

Value

A UMAP advanced analysis object

Slots

`clustering_channels` list the channels selected for the Dimensionality Reduction analysis, this can be either a list of short channel IDs (integer) OR long channel names (character)

`collapse_outliers` logical Dimension values that are significant outliers (z-score > 3) will be collapsed to be equal to the min or max value. Try this if you observe that most of the data appears squished within small region

`desired_events_per_file` numeric representing the number of desired events per file

`desired_total_events` numeric representing the number of desired total events per file

`event_sampling_method` character representing the name of event sampling method will be used, learn more about Event Sampling for Dimensionality Reduction analysis

`fcsfile_ids` list representing the fcs file ids

`gateset_id` numeric representing the selected gate id

`min_distance` numeric the effective minimum distance between embedded points, learn more about minimum distance for UMAP analysis

`num_neighbors` numeric the size of local neighborhood (in terms of number of neighboring sample points) used for manifold approximation, learn more about number of neighbors for UMAP analysis

`normalize_scales` logical representing whether or not to normalize scales

`users` *User Endpoints*

Description

Interact with user endpoints. One should never analyze alone...

Usage

```
## S4 method for signature 'UserSession'
users.list(
  UserSession,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
users.show(
  UserSession,
  user_id,
  output = "default",
  timeout = UserSession@short_timeout
)
```

Arguments

`UserSession`	Cytobank UserSession object
`output`	character representing the output format **[optional]** - *users.list, users.show :* `("default", "raw")`
`timeout`	integer representing the request timeout time in seconds **[optional]**
`user_id`	integer representing a user ID

Details

`users.list` List all users on a Cytobank server (admin access only). Outputs a dataframe [default] or raw list with all fields present.
- *Optional output parameter, specify one of the following:* `("default", "raw")`

`users.show` Show user details (admin access only, except for self). - *Optional output parameter, specify one of the following:* `("default", "raw")`

Examples

```
## Not run: # Authenticate via username/password
cyto_session <- authenticate(site="premium", username="cyril_cytometry", password="cytobank_rocks!")
# Authenticate via auth_token
cyto_session <- authenticate(site="premium", auth_token="my_secret_auth_token")

## End(Not run)
## Not run: # Dataframe of all users with all fields present
users.list(cyto_session)

# Raw list of all useres with all fields present
users.list(cyto_session, output="raw")

## End(Not run)
## Not run: users.show(cyto_session, user_id=2)
```

`UserSession-class` *S4 Cytobank UserSession Class*

Description

A Cytobank UserSession object that holds pertinent user information, used to make calls to various Cytobank endpoints. This class should never be called explicitly. If a user would like to create a new Cytobank UserSession object, utilize the authenticate function.

Value

A Cytobank UserSession object

Slots

`auth_token` character representing Cytobank user's authentication token (expires in 8 hours)

`long_timeout` numeric representing long request timeout times

`short_timeout` numeric representing short request timeout times

`site` character representing Cytobank user's site

`user_id` integer representing a Cytobank user's ID

Examples

```
cytobank_user <- new("UserSession", auth_token="my_auth_token", site="premium")
```

---

visne *viSNE Endpoints*

Description

Interact with viSNE advanced analyses using these endpoints.

Usage

```
## S4 method for signature 'UserSession,viSNE'
visne.copy_settings(
  UserSession,
  visne,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession,viSNE'
visne.delete(UserSession, visne, timeout = UserSession@short_timeout)

## S4 method for signature 'UserSession'
visne.list(
  UserSession,
  experiment_id,
  output = "default",
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession'
visne.new(
  UserSession,
  experiment_id,
  visne_name,
  timeout = UserSession@long_timeout
```

```
)

## S4 method for signature 'UserSession,viSNE'
visne.rename(
  UserSession,
  visne,
  visne_name,
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession,viSNE'
visne.run(
  UserSession,
  visne,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession'
visne.show(
  UserSession,
  experiment_id,
  visne_id,
  timeout = UserSession@short_timeout
)

## S4 method for signature 'UserSession,viSNE'
visne.status(
  UserSession,
  visne,
  output = "default",
  timeout = UserSession@long_timeout
)

## S4 method for signature 'UserSession,viSNE'
visne.update(UserSession, visne, timeout = UserSession@long_timeout)

visne.helper.set_populations(visne, population_id = NA, fcs_files = NA)
```

Arguments

`UserSession`	Cytobank UserSession object
`visne`	Cytobank viSNE object
`output`	character representing the output format **[optional]** - *visne.list, visne.run, visne.status :* (`"default"`, `"raw"`)
`timeout`	integer representing the request timeout time in seconds **[optional]**
`experiment_id`	integer representing an experiment ID

visne_name	character representing a new viSNE name
visne_id	integer representing a viSNE ID
population_id	integer representing a population **gate set ID**
fcs_files	vector/list of integers representing a list of FCS file IDs

Details

visne.copy_settings Copy viSNE advanced analysis settings from an experiment and returns a viSNE object.

visne.delete Delete a viSNE advanced analysis from an experiment.

visne.list List all viSNE advanced analyses from an experiment. Outputs a dataframe [default] or list with all fields present.
- *Optional output parameter, specify one of the following:* ("default", "raw")

visne.new Create a new viSNE advanced analysis from an experiment and returns a viSNE object.

visne.rename Rename a viSNE advanced analysis from an experiment and returns a viSNE object.

visne.run Run a viSNE advanced analysis from an experiment.

visne.show Show viSNE advanced analysis details from an experiment and returns a viSNE object.

visne.status Show the status of a viSNE advanced analysis from an experiment.

visne.update Update a viSNE advanced analysis from an experiment and returns the new viSNE object.

visne.helper.set_populations Set viSNE advanced analysis populations to be selected from an experiment and returns the new viSNE object with the new population selections. The population provided will be overwritten by the newly selected FCS files provided.

Examples

```
## Not run: # Authenticate via username/password
cyto_session <- authenticate(site="premium", username="cyril_cytometry", password="cytobank_rocks!")
# Authenticate via auth_token
cyto_session <- authenticate(site="premium", auth_token="my_secret_auth_token")

# cyto_visne refers to a viSNE object that is created from viSNE endpoints
#   examples: visne.new, visne.show (see details section for more)

## End(Not run)
## Not run: visne.copy_settings(cyto_session, visne=cyto_visne)

## Not run: visne.delete(cyto_session, visne=cyto_visne)

## Not run: # Dataframe of all viSNE advanced analyses with all fields present
visne.list(cyto_session, 22)

# Raw list of all viSNE advanced analyses with all fields present
visne.list(cyto_session, 22, output="raw")

## End(Not run)
## Not run: visne.new(cyto_session, 22, visne_name="My new viSNE analysis")
```

```
## Not run: visne.rename(cyto_session, visne=cyto_visne, visne_name="My updated viSNE name")

## Not run: visne.run(cyto_session, visne=cyto_visne)

## Not run: visne.show(cyto_session, 22, visne_id=2)

## Not run: visne.status(cyto_session, visne=cyto_visne)

## Not run: visne.update(cyto_session, visne=cyto_visne)

## Not run: visne.helper.set_populations(visne=cyto_visne, population_id=1, fcs_files=c(1,2,3))
```

`viSNE-class` *S4 viSNE Class*

Description

A viSNE object that holds pertinent viSNE analysis run information. This class should never be called explicitly. If a user would like to create a new Cytobank Dimensionality Reduction object, utilize the dimensionality_reduction.new function, or any other Dimensionality Reduction endpoints that return Dimensionality Reduction objects documented in the 'Details' section.

Value

A Dimensionality Reduction advanced analysis object

Slots

`iterations` numeric representing the number of times Dimensionality Reduction processes the dataset using its step-wise optimization algorithm, learn more about how iterations affect Dimensionality Reduction results

`perplexity` numeric representing a rough guess for the number of close neighbors any given cellular event will have, learn more about Dimensionality Reduction perplexity

`channels` list the channels selected for the Dimensionality Reduction analysis, this can be either a list of short channel IDs (integer) OR long channel names (character)

`compensation_id` the compensation ID selected for the Dimensionality Reduction analysis

`population_selections` dataframe representing which population(s) data will be sourced, learn more about selecting populations for Dimensionality Reduction

`sampling_total_count` numeric representing the total number of events to sample for the Dimensionality Reduction analysis

`sampling_target_type` character representing the event sampling type
- *choose one of the following :* `("proportional", "equal")`

`seed` character representing the seed, Dimensionality Reduction picks a random seed each run, but if users want reproducible data, setting the same seed will allow them to do this

`theta` numeric representing the balance of speed and accuracy in the Dimensionality Reduction run compared to the original tSNE algorithm, learn more about Dimensionality Reduction theta

`visne_id` numeric representing the Dimensionality Reduction analysis ID

Index

AdvancedAnalysis-class, 2
attachments, 3, *43*
authenticate, *64*
authenticate *(*authentication*)*, 6
authentication, 6
autogating, 7

citrus, 13
CITRUS endpoints that return CITRUS objects documented in the 'Details' section, *16*
CITRUS-class, 16
citrus.new, *16*
compensation, *60*
compensations, 17, *24*
CytobankAPI_news *(*news*)*, 41

Dimensionality Reduction endpoints that return Dimensionality Reduction objects documented in the 'Details' section, *19*, *61*, *68*
dimensionality_reduction, 20
dimensionality_reduction.new, *19*, *42*, *61*, *62*, *68*
DimensionalityReduction-class, 19
drop, 23

experiment, *4*, *10*, *14*, *18*, *21*, *24*, *31*, *35*, *39*, *43*, *45*, *48*, *50*, *51*, *56*, *59*, *66*
experiments, 24

FCS file, *27*, *60*
FCS files, *17*
fcs_files, 29
fcs_files.list, *3*, *19*
flowsom, 33
FlowSOM endpoints that return FlowSOM objects documented in the 'Details' section, *36*
FlowSOM-class, 36
flowsom.new, *36*

gates, *24*, 38, *48*, *60*

helper.channel_ids_from_long_names *(*helper_functions*)*, 40
helper.filter_names_to_ids_from_df *(*helper_functions*)*, 40
helper_functions, 40

news, 41

opt-SNE endpoints that return opt-SNE objects documented in the 'Details' section, *42*
optSNE-class, 42

panels, 43
panels.list, *3*, *19*, *41*
peacoqc, 44
PeacoQC endpoints that return PeacoQC objects documented in the 'Details' section, *47*
PeacoQC-class, 47
peacoqc.new, *47*
population, *38*, *60*
populations, 48, *60*
populations.list, *3*, *19*

sample_tags, 49
scales, *24*, 51
Show Experiment Details, *60*
spade, 52
SPADE endpoints that return SPADE objects documented in the 'Details' section, *58*
SPADE-class, 58
spade.new, *58*
statistics, 59

tSNE-class, 61

UMAP endpoints that return UMAP
        objects documented in the
        'Details' section, *62*
UMAP-class, 62
users, 63
UserSession-class, 64

visne, 65
viSNE-class, 68