

Package ‘ffcAPIClient’

November 6, 2020

Type Package

Title Functional Flows Calculator API Client

Version 0.9.7.4

Description A client for the Python-based functional flows calculator API hosted at eflows.ucdavis.edu. Also handles data processing and analysis related to the California Environmental Flows Framework (CEFF). Requires a token from the eflows.ucdavis.edu website to operate.

License MIT

Encoding UTF-8

LazyData true

Imports dplyr,
jsonlite,
httr,
uuid,
ggplot2,
dataRetrieval,
lubridate,
R6,
nhdplusTools (>= 0.3.15),
units,
tidyr,
data.table (>= 1.12.8),
futile.logger

Suggests testthat (>= 2.1.0),
knitr,
rmarkdown

RoxygenNote 7.1.0

VignetteBuilder knitr

R topics documented:

assess_alteration	2
attach_water_year_data	3
clean_account	3
early_or_late	4
evaluate_alteration	4

evaluate_gage_alteration	5
ffcAPIClient	7
FFCProcessor	7
fill_na_10th_percentile	9
filter_timeseries	9
flow_metrics	11
force_consistent_naming	11
gage_comids	12
get_comid_for_lon_lat	12
get_drh	12
get_ffc_parameters_for_comid	13
get_ffc_results_for_usgs_gage	13
get_predicted_flow_metrics	14
get_results_as_df	14
get_token	14
get_usgs_gage_data	15
merge_list	15
plot_drh	15
set_token	16
stream_class_data	16
USGSGage	17

Index 19

assess_alteration	<i>Assess hydrologic alteration by flow metric</i>
-------------------	--

Description

Returns a data frame with an alteration status assessment for every flow metric.

Usage

```
assess_alteration(percentiles, predictions, ffc_values, comid, annual)
```

Arguments

percentiles	dataframe of calculated FFC results percentiles, including the metric column and columns for p10,p25,p50,p75, and p90
predictions	dataframe of predicted flow metrics, as returned from get_predicted_flow_metrics.
ffc_values	dataframe of the raw results from the online FFC, as returned by evaluate_gage_alteration or get_results_as_df
comid	integer comid of the stream segment the previous parameters are for
annual	boolean indicating whether to run a year over year analysis. If TRUE, then the parameter percentiles changes and should be a data frame with only two columns - the first is still metric, but the second is just value representing the current year's value for the metric. predictions should then still have fields for the metric, p25, and p75, where p25 and p75 represent the lower and upper bounds for comparison, regardless of if they're calculated percentiles, or another set of bounds. When run in an annual mode, it assesses alteration similarly to the description above, and with the same result structure, but provides

likely_unaltered results when value is within the p25 and p75 values, and provides likely_altered otherwise without additional checks described in the CEFF guidance document, appendix F.

Details

Generates an alteration status assessment for every flow metric based on the rules developed under CEFF for flow alteration. This function pairs well with the boxplots for visualizing alteration, but only this function assesses the data under the rules. Returns a data frame with columns "metric", "status_code", "status", "alteration_type", "median_in_iqr", and "comid".

The comid will be the same for all rows, and will match what you provide as an input, but allows for merging of these results into larger tables.

status_code will be -1 (likely altered), 0 (indeterminate), 1 (likely unaltered), or NA (insufficient data to determine). status will be a text description of the status code (-1=likely_altered, 0=indeterminate, 1=likely_unaltered, NA=Not_enough_data). alteration_type will tell you the direction of potential alteration for likely altered and indeterminate metrics - the direction of alteration is determined by comparing the median value to the 25th and 7th percentiles of the predicted metrics. It will provide "low" or "high" values for most metrics and "early" or "late" values for timing metrics. For likely_unaltered metrics, it will provide "none_found" and for metrics with insufficient data, it will provide "undeterminable.". Also includes a boolean field median_in_iqr indicating whether the median is in the interquartile range.

attach_water_year_data

Add calendar_year/calendar_month/calendar_day/water_year fields

Description

Attaches fields for the year, month, day and water year to a data frame with a POSIX time field. Fields will be named calendar_year, calendar_month, calendar_day, and water_year.

Usage

```
attach_water_year_data(timeseries, date_field = "posix_time")
```

Arguments

timeseries.	A timeseries data frame with a date field
date_field.	A sortable field of date/time information - a good choice would be a field created by strptime or any other POSIX format time.

clean_account	<i>Clean account</i>
---------------	----------------------

Description

Deletes all runs in the online FFC for the user. Helpful in cases where the website configuration changes and makes accounts broken in the online interface.

Usage

```
clean_account(token)
```

Arguments

token. A valid token for the account to clean out. See install instructions for details

early_or_late	<i>Determine if timing metrics are early, late, or in range</i>
---------------	---

Description

Properly rolls over the calendar at 365 days, but can tell you if a metric is early, late, or "within range" based on the modeled early_value, modeled late_value, and the actual value.

Usage

```
early_or_late(value, early_value, late_value, days_in_water_year)
```

Details

It returns within range (0) if the value is between early_value and late_value. If not, it splits the distance between late_value and early_value in two, rolling over at the end of the calendar year, and assesses if the value is closer to the late_value (then returns late (1)), or the early value (then returns early (-1)).

This function is currently not used in the package - instead, a simpler evaluation that does not roll over the calendar year is used.

evaluate_alteration *Generate FFC Results and Plots for Timeseries Data*

Usage

```
evaluate_alteration(
    timeseries_df,
    token,
    comid,
    longitude,
    latitude,
    plot_output_folder,
    plot_results,
    date_format_string,
    return_processor
)
```

Arguments

timeseries_df	A timeseries dataframe that includes fields named "flow" and "date" for each record. Date should either be in MM/DD/YYYY format, or parameter date_format_string must be specified. The data frame may include other fields, which will be automatically dropped when sent to the FFC.
token	The token used to access the online FFC - see the Github repository's README under Setup for how to get this.
comid	The stream segment COMID where the gage is located. You may also have the package look this information up automatically based on longitude and latitude, but we discovered that our method for looking gage COMIDs up is error prone, and there is no authoritative dataset that relates gages to COMIDs correctly. It will be most accurate if you provide the comid yourself by looking it up (don't use nhdPlusTools with the latitude and longitude that's what we did that was error prone).
longitude	the longitude of the location the flow data were collected at.
latitude	the latitude of the location the flow data were collected at. If both longitude and latitude are defined, then and parameter comid is missing, then the COMID will be looked up. See notes on parameter comid for cautions and limitations.
plot_output_folder	Optional - when not provided, plots are displayed interactively only. When provided, they are displayed interactively and saved as files named by the functional flow component into the provided folder
plot_results	boolean, default TRUE - when TRUE, results are plotted to the screen and any folder provided. When FALSE, does no plotting.
date_format_string	character. Default " Processes timeseries data using the functional flows calculator and returns results for metric percentiles, annual metric values, predicted metric values, flow alteration, and drh data. See the documentation for evaluate_gage_alteration for complete details on the processing and what is returned.

```
evaluate_gage_alteration
```

Generate FFC Results and Plots for Gage Data

Description

This is a shortcut function that does most of the heavy lifting for you. Runs data through the FFC and transforms all results.

Usage

```
evaluate_gage_alteration(
  gage_id,
  token,
  comid,
  plot_output_folder,
  plot_results,
  force_comid_lookup,
  return_processor
)
```

Arguments

<code>gage_id</code>	The USGS gage ID to pull timeseries data from
<code>token</code>	The token used to access the online FFC - see the Github repository's README under Setup for how to get this.
<code>comid</code>	The stream segment COMID where the gage is located. In the past, the package looked this information up automatically but we discovered that our method for looking gage COMIDs up was error prone, and there is no authoritative dataset that relates gages to COMIDs. It will be most accurate if you provide the comid yourself by looking it up (don't use <code>nhdPlusTools</code> with the latitude and longitude - that's what we did that was error prone). You can re-enable the lookup behavior setting the <code>force_comid_lookup</code> parameter to <code>TRUE</code> .
<code>plot_output_folder</code>	Optional - when not provided, plots are displayed interactively only. When provided, they are displayed interactively and saved as files named by the functional flow component into the provided folder
<code>plot_results</code>	boolean, default <code>TRUE</code> - when <code>TRUE</code> , results are plotted to the screen and any folder provided. When <code>FALSE</code> , does no plotting.
<code>force_comid_lookup</code>	default <code>FALSE</code> . When <code>TRUE</code> , the COMID for the segment will be automatically looked up based on the latitude and longitude. This method is error prone and it is advised you leave it off. Where an error is known, the package corrects the COMID based on an internal list of gage/comid pairs (eg: Jones Bar on the Yuba River). It is recommended you leave this as <code>FALSE</code> and look up the comid yourself to ensure that you choose the correct mainstem or tributary near stream junctions, but if you need to bulk process data, this parameter is available to retrieve COMIDs.

Details

If you provide it a USGS gage ID and your token to access the online functional flows calculator, this function then:

1. Download the timeseries data for the USGS gage
2. Look up the predicted unimpaired metric values for the gage's stream segment
3. Send the timeseries data through the functional flows calculator
4. Transform the results into a data frame with rows for years and metric values as columns
5. Produce percentiles for those metric values using R's recommended quantile method type 7 (which may return differing results from other methods, Excel, etc)
6. Transform the dimensionless reference hydrograph data into a data frame
7. Determines the alteration by flow metric for the observed versus predicted values
8. Output plots comparing the observed timeseries data with the predicted unimpaired metric values.

Items 4, 5, 6, and 7 are returned back to the caller as a list with keys "ffc_results", "ffc_percentiles", "drh_data", and "alteration" for any further processing. The list also includes "predicted_percentiles", with the predicted flow metrics for the segment.

 ffcAPIClient

ffcAPIClient: Processes time-series flow data using the online functional flows calculator

Description

For now, see the documentation for [evaluate_alteration](#) and [evaluate_gage_alteration](#)

Examples

```
## Not run:
# If you have a gage and a token, you can get all results simply by running
ffcAPIClient::evaluate_gage_alteration(gage_id = 11427000, token = "your_token", plot_output_folder = "C:/Use
# output_folder is optional. When provided, it will save plots there. It will show plots regardless.

# If you have a data frame with flow and date fields that isn't a gage, you can run
ffcAPIClient::evaluate_alteration(timeseries_df = your_df, token = "your_token", plot_output_folder = "C:/Use
# it also *REQUIRES* you provide either a comid argument with the stream segment COMID, or both
# longitude and latitude arguments.
# Make sure that dates are in the same format as the FFC requires on its website. We may add reformatting in the f

## End(Not run)
```

FFCProcessor	<i>FFCProcessor Class</i>
--------------	---------------------------

Description

The new workhorse of the client - this class is meant to bring together the scattershot functions in other parts of the package so that data can be integrated into a single class with a single set of tasks. Other functions are likely to be supported for a while (and this may even rely on them), but long run, much of the code in this file might move into this class, with the shortcut functions creating this class behind the scenes and returning an instance of this object.

Details

More details to come, and more examples. For now, still use the general functions [evaluate_alteration](#) and [evaluate_gage_alteration](#)

Get Gage ID for FFCProcessor

We may not always have a gage ID, but may want to just get one if it exists - this function gets a gage ID if we're using a gage or returns NA otherwise

Provides alteration scores

Checks the results against the predictions and returns the appropriate alteration score

Methods

Public methods:

- `FFCProcessor$set_up()`
- `FFCProcessor$run()`
- `FFCProcessor$rename_inconsistent_metrics()`
- `FFCProcessor$get_gage_id()`
- `FFCProcessor$step_one_functional_flow_results()`
- `FFCProcessor$step_two_explore_ecological_flow_criteria()`
- `FFCProcessor$step_three_assess_alteration()`
- `FFCProcessor$get_ffc_results()`
- `FFCProcessor$evaluate_alteration()`
- `FFCProcessor$clone()`

Method `set_up()`:

Usage:

```
FFCProcessor$set_up(gage_id, timeseries, comid, token)
```

Method `run()`:

Usage:

```
FFCProcessor$run()
```

Method `rename_inconsistent_metrics()`:

Usage:

```
FFCProcessor$rename_inconsistent_metrics()
```


Method get_gage_id():*Usage:*

FFCProcessor\$get_gage_id()

Method step_one_functional_flow_results():*Usage:*

```
FFCProcessor$step_one_functional_flow_results(
  gage_id,
  timeseries,
  comid,
  token,
  output_folder
)
```

Method step_two_explore_ecological_flow_criteria():*Usage:*

FFCProcessor\$step_two_explore_ecological_flow_criteria()

Method step_three_assess_alteration():*Usage:*

FFCProcessor\$step_three_assess_alteration()

Method get_ffc_results():*Usage:*

FFCProcessor\$get_ffc_results()

Method evaluate_alteration():*Usage:*

FFCProcessor\$evaluate_alteration()

Method clone(): The objects of this class are cloneable with this method.*Usage:*

FFCProcessor\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

fill_na_10th_percentile

Fill 10th Percentile NA values when 25th percentile value is 0

Description

Sometimes data from the predicted metrics API has NA values in the 10th percentile field due to modeling errors where these numbers were originally set to negative values and were replaced with NA in the API. This function, which needs to be enabled using the `ffc$predicted_percentiles_fill_na_p10` flag on FFCProcessor objects, fills any NA values it finds in the p10 field *if* the p25 field is 0. Otherwise, it leaves them as they are. Raises a warning if it finds any NA values in the p10 field regardless of whether it fills them.

Usage

```
fill_na_10th_percentile(df, fill_na_p10)
```

Details

This function can be used with any other data frame that contains field p10 and p25 as well, though I'm not sure the conditions you'd need to!

filter_timeseries	<i>Filter Timeseries data</i>
-------------------	-------------------------------

Description

When we send data to the FFC, it needs to meet a specific set of requirements. The code in this file aims to make sure input data, whether gaged or a manual timeseries, follows those requirements. Namely:

Usage

```
filter_timeseries(
  timeseries,
  date_field,
  flow_field,
  date_format_string,
  max_missing_days = 7,
  max_consecutive_missing_days = 1,
  fill_gaps = "no"
)
```

Arguments

timeseries.	A timeseries data frame with date and flow fields
date_field.	character vector/string with the name of the field that holds the dates
flow_field.	character vector/string with the name of the field that holds flow values
date_format_string.	A format string to use when parsing dates into POSIX time
max_missing_days.	How many days can be missing from each water year before it is considered too incomplete and will be dropped? The water year will need to have <i>*more*</i> missing days than this value (so, the default of 7 means that a water year with 7 missing values will be kept, but a water year with 8 missing values will be dropped).
max_consecutive_missing_days.	How many days in a row can be missing before a water year is dropped? This is evaluated independently from max_missing_days, so a single failure of the rule attached to this parameter causes the water year to be dropped. The previous parameter handles the total number of missing values across a water year, while this parameter only looks at each gap's length. A max_consecutive_missing_days value of 1 means each gap can only be a single day in length (so, if we had values for day 1 and 2, were missing day 3, and then had values for day 4 and 5,

that'd be acceptable. If we were also missing day 2 or 4, that would cause the water year to be dropped).

`fill_gaps.` Currently unimplemented, but would allow for filling remaining gaps after water the rules are evaluated according to the other parameters. Defaults to "no", which turns it off. In the future, we expect to add two other options here: "linear" to linearly interpolate across gaps and "previous" to fill gaps with the closest valid value before the gap.

Details

1. We should only send complete water years - partial water years at the beginning or end have an outsize influence on the calculations. This rule will actually be handled just by adhering to the next two rules.
2. Along the same lines, we shouldn't allow any large gaps. When we find a gap of *2 or more days*, then we should drop the entire water year.
3. We should only allow 7 total missing days, which we will fill either with values from the previous day, or a linear interpolation (maybe with a flag?) before sending to the FFC.

This function accepts a timeseries data frame, assesses/filters it according to these rules, then returns a new timeseries to the caller. A record is considered a gap if it is missing for a date, or if it is present, but has a flow value of NA. The timeseries should already be *daily* data.

This function is also staged to fill any remaining gaps in preparation for the online FFC disabling that functionality, but the code has not yet been enabled here.

flow_metrics

Modeled flow metric predictions for all stream segments

Description

Contains the 10th, 25th, 50th, 75th, and 90th percentile values for each flow metric and stream segment combination. It is a data frame where the metrics are rows with names in the `Metric` field, stream segment ID is in the `COMID` field and percentiles are available as fields such as `pct_10`, `pct_25`, etc for each percentile.

Usage

`flow_metrics`

Format

A data frame :

name text

name text ...

<https://github.com/ceff-tech/>

force_consistent_naming

Set Preference to Rename Metrics Consistently

Description

The Peak Magnitude flow metrics are named Peak_2, Peak_5, and Peak_10, while other flow metrics for magnitude use a Component_Mag format. Setting this preference forces all peak metrics to use the naming Peak_Mag_2, Peak_Mag_5, and Peak_Mag_10. This is inconsistent with CEFF, but internally consistent with metric names. The default behavior is not to do this.

Usage

```
force_consistent_naming(force_to)
```

Arguments

force_to boolean Set to TRUE to enable metric renaming. Set to FALSE to disable it.

Details

To use, run force_consistent_naming(TRUE). Then, any function that returns metrics from a FFCProcessor object (includes the FFCProcessor itself and all evaluate_alteration functions - in the future should be any exported function) will rename the peak metrics. To turn off, run force_consistent_naming(FALSE).

gage_comids

Where we know the lat/long will produce the wrong COMID for a gage (such as giving us a nearby tributary and not the mainstem, or vice versa), we can hardcode their COMIDs here to make sure we get the correct location. We haven't looked through all gages to make sure every one is correct, but as we find them, this will help improve results and reduce error

Description

Where we know the lat/long will produce the wrong COMID for a gage (such as giving us a nearby tributary and not the mainstem, or vice versa), we can hardcode their COMIDs here to make sure we get the correct location. We haven't looked through all gages to make sure every one is correct, but as we find them, this will help improve results and reduce error

Usage

```
gage_comids
```

Format

An object of class list of length 1.

get_comid_for_lon_lat *Retrieves COMID for a given USGS gage which collects daily data.*

Description

This function returns the COMID associated with a specific USGS gage. It can be used to associate gage data with flow metric predictions a stream segment identified with the com_id input variable.

Usage

```
get_comid_for_lon_lat(longitude, latitude)
```

Arguments

longitude	numeric. Longitude or X.
latitude	numeric. Longitude or Y.

get_drh *Returns the dimensionless reference hydrograph results as a data frame*

Description

Returns the dimensionless reference hydrograph results as a data frame

Usage

```
get_drh(results)
```

get_ffc_parameters_for_comid *Get the parameters sent to the FFC for a stream segment*

Description

Given a COMID, looks up the hydrogeomorphic stream classification, then uses that to find the default parameters that should be sent to the FFC online for that stream class. Returns a nested list of parameters to send to the FFC.

Usage

```
get_ffc_parameters_for_comid(comid)
```

Arguments

comid	An NHD stream segment COMID
-------	-----------------------------

```
get_ffc_results_for_usgs_gage
```

Run Gage Data Through the Functional Flows Calculator

Description

Provided with an integer Gage ID, this function pulls the timeseries data for the gage and processes it in a single step. Returns the functional flow calculator's results list.

Usage

```
get_ffc_results_for_usgs_gage(gage_id)
```

Arguments

gage_id	integer. The USGS Gage ID value for the gage you want to return timeseries data for
---------	---

Value

list. Functional Flow Calculator results

```
get_predicted_flow_metrics
```

Retrieves flow predicted flow metric values for a stream segment

Description

This function returns the 10th, 25th, 50th, 75th, and 90th percentile values for each flow metric as predicted for the stream segment you identify with the comid input variable. It returns a data frame where the metrics are rows with names in the metric field, and percentiles are available as fields such as pct_10, pct_25, etc for each percentile.

Usage

```
get_predicted_flow_metrics(comid, online, wyt, fill_na_p10)
```

Arguments

comid	character. A string of a NHD COMID to retrieve metrics for.
online	boolean. Default FALSE. When TRUE, retrieves data from TNC's experimental predicted flow metrics API. When FALSE, uses internal data to pull flow metrics. Both are reasonably fast, but offline is good for reliability, but may end up using older data. Online should pull the most current data if there are updates. FALSE is the default largely because the API is still unstable.
wyt	character. When online = TRUE, filters the result to records with only the specific water year type indicated. See TNC's flow API documentation at flow-api.codefornature.org for options. If you want the records to come back unfiltered, use "any", and for the non-WYT records, use "all" (it's a specific keyword the data uses - not our choice - sorry for any confusion!).

get_results_as_df	<i>Convert FFC results list to data frame with metric names</i>
-------------------	---

Description

More documentation forthcoming

Usage

```
get_results_as_df(results, drop_fields)
```

get_token	<i>Retrieve Previously Set Token</i>
-----------	--------------------------------------

Description

Retrieves the authorization token previously set by set_token in the same R session.

Usage

```
get_token()
```

get_usgs_gage_data	<i>Retrieves USGS timeseries gage data</i>
--------------------	--

Description

This is just a helper function that calls the gage constructor, gets the flows and returns them in one step. Useful in situations where we don't need the flexibility of the USGSGage class

Usage

```
get_usgs_gage_data(gage_id)
```

Arguments

gage_id	integer. The USGS Gage ID value for the gage you want to return timeseries data for
---------	---

Value

dataframe. Will include a flow field (CFS) and a date field (MM/DD/YYYY)

merge_list	<i>Merges Data Frames by Year Column</i>
------------	--

Description

Just a simple function that can be used with Reduce to merge multiple data frames together by year

Usage

```
merge_list(df1, df2)
```

plot_drh	<i>Plots the Dimensionless Reference Hydrograph</i>
----------	---

Description

Given a set of results data from get_ffc_results_for_df or get_ffc_results_for_usgs_gage, processes the DRH data and returns a plot object.

Usage

```
plot_drh(results, output_path = NULL)
```

Arguments

results	list.
output_path,	default NULL. Optional. When set, saves the DRH plot to the output file path provided.

Details

Credit to Ryan Peek for the plotting code in this function.

set_token	<i>Set Eflows Website Access Token</i>
-----------	--

Description

Provide the token string used for accessing the Eflows site. A token is a method of authorization for identifying your user account within scripts. By providing the token, this package uses your user account when interacting with the eflows web service/API.

Usage

```
set_token(token_string)
```

Arguments

token_string	character
--------------	-----------

stream_class_data	<i>Geomorphic Stream Classification</i>
-------------------	---

Description

Contains the geomorphic classification by stream COMID for ~70,000 stream segments in California (low-order streams excluded). Streams were classified as described in Lane, Belize A., Samuel Sandoval-Solis, Eric D. Stein, Sarah M. Yarnell, Gregory B. Pasternack, and Helen E. Dahlke. 2018. “Beyond Metrics? The Role of Hydrologic Baseline Archetypes in Environmental Water Management.” *Environmental Management* 62 (4): 678–93. <https://doi.org/10.1007/s00267-018-1077-7>.

Usage

```
stream_class_data
```

Format

A data frame :

CLASS The stream classification ID

COMID The NHD COMID of the stream segment

CLASS_CODE The character stream classification ID - follows the form: SM = Snowmelt, HSR = High Volume Snowmelt and Rain, LSR = Low Volume Snowmelt and Rain, WS = Winter Storms, GW = Groundwater, PGR = Perennial Groundwater and Rain, FER = Flashy Ephemeral Rain, RGW = Rain and Seasonal Groundwater, HLP = High elevation, low precipitation

<https://doi.org/10.1007/s00267-018-1077-7>

USGSGage	<i>USGS Gage Retrieval Tools</i>
----------	----------------------------------

Description

This class retrieves data for a USGS gage.

Methods

Public methods:

- `USGSGage$validate()`
- `USGSGage$get_data()`
- `USGSGage$get_comid()`
- `USGSGage$get_predicted_metrics()`
- `USGSGage$clone()`

Method `validate()`:

Usage:

```
USGSGage$validate(latlong)
```

Details: Validates that gage is ready to run requests

Internal method. Checks parameters to make sure they're ready for other methods on the object.

Method `get_data()`:

Usage:

```
USGSGage$get_data()
```

Method `get_comid()`:

Usage:

```
USGSGage$get_comid()
```

Details: Looks up the COMID for this gage

This method looks up the COMID for the gage and sets the comid attribute. It does not return the COMID. It returns this object for chaining. The gage's id, latitude, and longitude attributes must be set before running this. latitude and longitude can be set manually, or by running `get_data()`. Can be error prone near stream junctions. If you have the means to get a reliable COMID for a gage, do so - in this method, we look up the stream segment by long/lat using `nhdPlusTools`.

Method `get_predicted_metrics()`:

Usage:

```
USGSGage$get_predicted_metrics(force_comid_lookup)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
USGSGage$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
#library(ffcAPIClient)
#gageid <- 11427000
#gage <- USGSGage$new()
#gage$id <- gageid
#gage$get_data()
#gage$get_comid()
#gage$comid
#14996611
#ffcAPIClient::get_predicted_flow_metrics(gage$comid)
```

	metric	COMID	p10	p25	p50	p75	p90	source
DS_Dur_WS	14996611	1.051875e+02	1.273438e+02	154.0625	1.785563e+02	1.953908e+02	model	
DS_Mag_50	14996611	4.998793e+01	6.732828e+01	104.4028	1.464183e+02	1.882733e+02	model	
DS_Mag_90	14996611	9.314097e+01	1.291930e+02	173.6844	2.382053e+02	3.393799e+02	model	
DS_Tim	14996611	2.720000e+02	2.823875e+02	296.8875	3.070000e+02	3.210167e+02	model	
FA_Dur	14996611	2.000000e+00	3.000000e+00	4.0000	6.000000e+00	8.000000e+00	obs	
FA_Mag	14996611	1.294269e+02	1.886283e+02	289.6838	4.540329e+02	8.514823e+02	model	
FA_Tim	14996611	7.816667e+00	1.400000e+01	24.6250	2.900000e+01	4.217000e+01	model	
Peak_10	14996611	1.243107e+04	1.947545e+04	22830.3355	3.124928e+04	3.767889e+04	model	
Peak_20	14996611	8.078893e+03	1.227363e+04	20218.4829	2.087196e+04	2.087196e+04	model	
Peak_50	14996611	3.532988e+03	7.350986e+03	8542.1191	8.969386e+03	8.969386e+03	model	
Peak_Dur_10	14996611	1.000000e+00	1.000000e+00	1.0000	2.000000e+00	4.000000e+00	obs	
Peak_Dur_20	14996611	1.000000e+00	1.000000e+00	2.0000	3.000000e+00	6.000000e+00	obs	
Peak_Dur_50	14996611	1.000000e+00	1.000000e+00	4.0000	1.000000e+01	2.900000e+01	obs	
Peak_Fre_10	14996611	1.000000e+00	1.000000e+00	1.0000	1.000000e+00	2.000000e+00	obs	

Peak_Fre_20	14996611	1.000000e+00	1.000000e+00	1.0000	2.000000e+00	3.000000e+00	obs
Peak_Fre_50	14996611	1.000000e+00	1.000000e+00	2.0000	3.000000e+00	5.000000e+00	obs
SP_Dur	14996611	4.700000e+01	5.900000e+01	72.0000	9.527500e+01	1.215417e+02	model
SP_Mag	14996611	1.067727e+03	1.662598e+03	2489.0563	3.771512e+03	5.809320e+03	model
SP_ROC	14996611	3.845705e-02	4.863343e-02	0.0625	8.132020e-02	1.141117e-01	obs
SP_Tim	14996611	1.607717e+02	1.905000e+02	218.7500	2.354750e+02	2.447583e+02	model
Wet_BFL_Dur	14996611	7.633333e+01	1.073000e+02	141.1958	1.633750e+02	1.875000e+02	model
Wet_BFL_Mag_10	14996611	1.519943e+02	1.960031e+02	278.2581	4.384614e+02	5.489183e+02	model
Wet_BFL_Mag_50	14996611	4.148992e+02	5.902507e+02	924.1728	1.175461e+03	1.426576e+03	model
Wet_Tim	14996611	4.937500e+01	5.905000e+01	73.0000	8.835625e+01	1.035083e+02	model

Index

* datasets

- flow_metrics, [11](#)
- gage_comids, [12](#)
- stream_class_data, [16](#)

assess_alteration, [2](#)
attach_water_year_data, [3](#)

clean_account, [3](#)

early_or_late, [4](#)
evaluate_alteration, [4](#), [7](#)
evaluate_gage_alteration, [5](#), [7](#)

ffcAPIClient, [7](#)
FFCProcessor, [7](#)
fill_na_10th_percentile, [9](#)
filter_timeseries, [9](#)
flow_metrics, [11](#)
force_consistent_naming, [11](#)

gage_comids, [12](#)
get_comid_for_lon_lat, [12](#)
get_drh, [12](#)
get_ffc_parameters_for_comid, [13](#)
get_ffc_results_for_usgs_gage, [13](#)
get_predicted_flow_metrics, [14](#)
get_results_as_df, [14](#)
get_token, [14](#)
get_usgs_gage_data, [15](#)

merge_list, [15](#)

plot_drh, [15](#)

set_token, [16](#)
stream_class_data, [16](#)

USGSGage, [17](#)