

# Package ‘ffcAPIClient’

February 13, 2020

**Type** Package

**Title** Functional Flows Calculator API Client

**Version** 0.9.5.8

**Description** A client for the Python-based functional flows calculator API hosted at [eflows.ucdavis.edu](http://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](http://eflows.ucdavis.edu) website to operate.

**License** MIT

**Encoding** UTF-8

**LazyData** true

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

**Suggests** testthat (>= 2.1.0)

**RoxygenNote** 7.0.2

## R topics documented:

assess_alteration . . . . .	2
early_or_late . . . . .	3
evaluate_alteration . . . . .	3
evaluate_gage_alteration . . . . .	4
ffcAPIClient . . . . .	6
FFCProcessor . . . . .	6
flow_metrics . . . . .	8
gage_comids . . . . .	8
get_comid_for_lon_lat . . . . .	9
get_drh . . . . .	9

get_ffc_parameters_for_comid . . . . .	9
get_ffc_results_for_usgs_gage . . . . .	10
get_predicted_flow_metrics . . . . .	10
get_results_as_df . . . . .	11
get_token . . . . .	11
get_usgs_gage_data . . . . .	11
merge_list . . . . .	12
plot_drh . . . . .	12
set_token . . . . .	12
stream_class_data . . . . .	13
USGSGage . . . . .	13
<b>Index</b>	<b>15</b>

---

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.

---

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	<i>Generate FFC Results and Plots for Timeseries Data</i>
---------------------	---

---

### Description

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.

**Usage**

```

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

```

**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.

**Details**

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  
)
```

## Arguments

gage_id	The USGS gage ID to pull timeseries data from
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. 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 nhdPlusTools with the latitude and longitude - that's what we did that was error prone). You can re-enable the lookup behavior setting the force_comid_lookup parameter to TRUE.
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.
force_comid_lookup	default FALSE. When TRUE, 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 FALSE 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	<i>ffcAPIClient: Processes time-series flow data using the online functional flows calculator</i>
--------------	---

---

## 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](#)

## Methods

### Public methods:

- `FFCProcessor$set_up()`
- `FFCProcessor$run()`
- `FFCProcessor$generate_functional_flow_results()`
- `FFCProcessor$explore_ecological_flow_criteria()`
- `FFCProcessor$assess_alteration()`
- `FFCProcessor$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 `generate_functional_flow_results()`:

*Usage:*

```
FFCProcessor$generate_functional_flow_results()
```

### Method `explore_ecological_flow_criteria()`:

*Usage:*

```
FFCProcessor$explore_ecological_flow_criteria()
```

### Method `assess_alteration()`:

*Usage:*

```
FFCProcessor$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.

---

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/>

---

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	<i>Returns the dimensionless reference hydrograph results as a data frame</i>
---------	---

---

### Description

Returns the dimensionless reference hydrograph results as a data frame

### Usage

```
get_drh(results)
```

---

get_ffc_parameters_for_comid	<i>Get the parameters sent to the FFC for a stream segment</i>
------------------------------	--

---

### 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)
```

### 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 <a href="http://flow-api.codefornature.org">flow-api.codefornature.org</a> 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)
```

**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 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.

## Details

```
#library(ffcAPIClient) #gageid <- 11427000 #gage <- USGSGage$new() #gage$gid <- gageid #gage$get_data()
#gage$get_comid() #gage$comid [1] 14996611 #ffcAPIClient::get_predicted_flow_metrics(gage$comid)
metric COMID p10 p25 p50 p75 p90 source 70804 DS_Dur_WS 14996611 1.051875e+02 1.273438e+02
154.0625 1.785563e+02 1.953908e+02 model 211050 DS_Mag_50 14996611 4.998793e+01 6.732828e+01
104.4028 1.464183e+02 1.882733e+02 model 351296 DS_Mag_90 14996611 9.314097e+01 1.291930e+02
173.6844 2.382053e+02 3.393799e+02 model 491542 DS_Tim 14996611 2.720000e+02 2.823875e+02
296.8875 3.070000e+02 3.210167e+02 model 586665 FA_Dur 14996611 2.000000e+00 3.000000e+00
4.0000 6.000000e+00 8.000000e+00 obs 702508 FA_Mag 14996611 1.294269e+02 1.886283e+02
289.6838 4.540329e+02 8.514823e+02 model 842754 FA_Tim 14996611 7.816667e+00 1.400000e+01
24.6250 2.900000e+01 4.217000e+01 model 983000 Peak_10 14996611 1.243107e+04 1.947545e+04
22830.3355 3.124928e+04 3.767889e+04 model 1123246 Peak_20 14996611 8.078893e+03 1.227363e+04
```

```

20218.4829 2.087196e+04 2.087196e+04 model 1263492 Peak_50 14996611 3.532988e+03 7.350986e+03
8542.1191 8.969386e+03 8.969386e+03 model 1358615 Peak_Dur_10 14996611 1.000000e+00
1.000000e+00 1.0000 2.000000e+00 4.000000e+00 obs 1429335 Peak_Dur_20 14996611 1.000000e+00
1.000000e+00 2.0000 3.000000e+00 6.000000e+00 obs 1500055 Peak_Dur_50 14996611 1.000000e+00
1.000000e+00 4.0000 1.000000e+01 2.900000e+01 obs 1570775 Peak_Fre_10 14996611 1.000000e+00
1.000000e+00 1.0000 1.000000e+00 2.000000e+00 obs 1641495 Peak_Fre_20 14996611 1.000000e+00
1.000000e+00 1.0000 2.000000e+00 3.000000e+00 obs 1712215 Peak_Fre_50 14996611 1.000000e+00
1.000000e+00 2.0000 3.000000e+00 5.000000e+00 obs 1828058 SP_Dur 14996611 4.700000e+01
5.900000e+01 72.0000 9.527500e+01 1.215417e+02 model 1968304 SP_Mag 14996611 1.067727e+03
1.662598e+03 2489.0563 3.771512e+03 5.809320e+03 model 2063427 SP_ROC 14996611 3.845705e-
02 4.863343e-02 0.0625 8.132020e-02 1.141117e-01 obs 2179270 SP_Tim 14996611 1.607717e+02
1.905000e+02 218.7500 2.354750e+02 2.447583e+02 model 2319516 Wet_BFL_Dur 14996611
7.633333e+01 1.073000e+02 141.1958 1.633750e+02 1.875000e+02 model 2459762 Wet_BFL_Mag_10
14996611 1.519943e+02 1.960031e+02 278.2581 4.384614e+02 5.489183e+02 model 2600008
Wet_BFL_Mag_50 14996611 4.148992e+02 5.902507e+02 924.1728 1.175461e+03 1.426576e+03
model 2740254 Wet_Tim 14996611 4.937500e+01 5.905000e+01 73.0000 8.835625e+01 1.035083e+02
model

```

## Methods

### Public methods:

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

### Method `validate()`:

*Usage:*

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

### Method `get_data()`:

*Usage:*

```
USGSGage$get_data()
```

### Method `get_comid()`:

*Usage:*

```
USGSGage$get_comid()
```

### 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.

# Index

## \*Topic **datasets**

- flow\_metrics, [8](#)
- gage\_comids, [8](#)
- stream\_class\_data, [13](#)

assess\_alteration, [2](#)

early\_or\_late, [3](#)

evaluate\_alteration, [3](#), [6](#)

evaluate\_gage\_alteration, [4](#), [6](#)

ffcAPIClient, [6](#)

FFCProcessor, [6](#)

flow\_metrics, [8](#)

gage\_comids, [8](#)

get\_comid\_for\_lon\_lat, [9](#)

get\_drh, [9](#)

get\_ffc\_parameters\_for\_comid, [9](#)

get\_ffc\_results\_for\_usgs\_gage, [10](#)

get\_predicted\_flow\_metrics, [10](#)

get\_results\_as\_df, [11](#)

get\_token, [11](#)

get\_usgs\_gage\_data, [11](#)

merge\_list, [12](#)

plot\_drh, [12](#)

set\_token, [12](#)

stream\_class\_data, [13](#)

USGSGage, [13](#)