Package ‘ffcAPIClient’

January 14, 2020

Type Package

Title Functional Flows Calculator API Client

Version 0.9.4

Author Nick Santos, Ryan Peek

Maintainer Nick Santos <nrsantos@ucdavis.edu>

Description A client for the Python-based functional flows calculator API hosted
at eflows.ucdavis.edu. Requires a token from the eflows.ucdavis.edu website to
operate. More information forthcoming. See README at https://github.com/ceff-tech/ffc_api_client
Use four spaces when indenting paragraphs within the Description.

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:

`determine_status` *Calculate the alteration status of a flow metric*

Description

This method returns an alteration status record for a specific flow metric, but requires the calculated FFC percentiles, a lower and upper bound, and a set of observations that have already been assessed for whether they're within that lower or upper bound so that they are -1 for low/early, 0 for within range, and 1 for high/late. They need to already be assessed because some metrics (*ahem* timing) need their own ways to assess low/high, or early/late

Usage

```
determine_status(
  percentiles,
  low_bound,
  high_bound,
  assessed_observations,
  metric,
  days_in_water_year,
  prediction_proportion
)
```

Arguments

`percentiles`	data frame row - should have a named value "p50" that can be accessed, at the very least. These are the calculated percentile values from the FFC.
`low_bound`	a value that is the lower end of the normal range for this metric - typically the p10 value from predicted metrics
`high_bound`	a value that is the upper end of the normal range for this metric - typically the p90 value from predicted metrics

`assessed_observations`	vector of raw observed metric values (FFC output) that has already been assessed for whether it is in range so that records that are low/early are -1, records that are in range are 0, and records that are high/late are 1
`metric`	character name of the metric - case sensitive. Currently only used for timing metrics, which must have "_Tim" in the name
`days_in_water_year`	numeric of how many days in the water year (typically 365, but could be 366).
`predicted_proportion`	numeric. When we know that we're not unaltered, we construct an interval to assess if we're altered, which is a two-sided multiplication of the low_bound and the high_bound by (1+prediction_proportion). Typically 0.2

`early_or_late` *Determine if timing metrics are early, late, or in range*

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. 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))

Usage

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

`evaluate_alteration` *Generate FFC Results and Plots for Timeseries Data*

Description

Generate FFC Results and Plots for Timeseries Data

Usage

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

`evaluate_gage_alteration` *Generate FFC Results and Plots for Gage Data*

Description

Generate FFC Results and Plots for Gage Data

Usage

```
evaluate_gage_alteration(gage_id, token, plot_output_folder)
```

`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` *FFCProcessor Class*

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$get_ffc_results()`
- `FFCProcessor$evaluate_alteration()`
- `FFCProcessor$clone()`

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/

`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_comid_for_usgs_gage` *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_usgs_gage(gage_id)
```

Arguments

`gage_id`	character. A character formatted 8 digit USGS Gage ID.

`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_results_for_df

Run Data Frame Through Functional Flows Calculator

Description

This is primarily an internal function used to run data through the functional flows calculator online, but is also available for those that wish to run the data themselves and then do any other handling and transformation for postprocessing on their own.

Usage

```
get_ffc_results_for_df(flows_df, flow_field, date_field, start_date)
```

Arguments

`flows_df`	DataFrame. A time series data frame with flow and date columns
`flow_field`	character, default "flow". The name of the field in `df` that contains flow values.
`date_field`	character, default "date". The name of the field in `df` that contains date values for each flow. The date field must be in MM/DD/YYYY format as either factor or character values - true dates likely will not work based on the API we're using. If you need to convert date values, add a field to your existing data frame with the values in MM/DD/YYYY format before providing it to this function.
`start_date`	character, default "10/1". What month and day should the water year start on? Neither month nor day needs to be zero-padded here, so March first could just be 3/1, while December 12th can be 12/12.

Details

Most people will want to use `evaluate_alteration` (for timeseries dataframes) or `evaluate_gage_alteration` (for USGS gages) instead.

Internally, this is the primary function to use from the API client itself to obtain raw FFC results. It will generate a unique ID, run the data frame through the FFC, and then delete the results for that ID from the website so as not to clutter up the user's account, or store too much data on the server side.

Value

list of results from the functional flows calculator. More information will be forthcoming as we inspect the structure of what is returned.

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, start_date)
```

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 `com_id` 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(com_id)
```

Arguments

com_id	character. A string of a NHD COMID to retrieve metrics for.

get_results_as_df *Convert FFC results list to data frame with metric names*

Description

More documentation forthcoming

Usage

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

`get_results_for_name` *Retrieve processed results from FFC.*

Description

Gets the results for the given named run of the FFC. Returns the nested list - all other processing must be handled by the caller.

Usage

```
get_results_for_name(name, autodelete)
```

Arguments

`name`	the name of the run to retrieve from the online FFC
`autodelete`	when TRUE, deletes the run in the online FFC, if found. When FALSE, leaves run in FFC online for later retrieval.

`get_token` *Retrieve Previously Set Token*

Description

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

Usage

```
get_token()
```

`get_usgs_gage_data` *Retrieves USGS timeseries gage data*

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` *Merges Data Frames by Year Column*

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` *Plots the Dimensionless Reference Hydrograph*

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.

`process_data` *Send flow data for processing*

Description

In most cases, you won't need to use this function! If you're wondering what to do, use get_ffc_results_for_df instead.

Usage

```
process_data(flows_df, flow_field, date_field, start_date, name)
```

Details

Sends flow timeseries data off to the functional flows calculator. Does not retrieve results!

`set_token` *Set Eflows Website Access Token*

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

`single_metric_alteration` *Assess the alteration of a single flow metric*

Description

Given a metric's calculated percentiles, raw FFC output values, and predictions, returns a row of information indicating whether or not that metric is likely altered, indeterminate, or likely unaltered. Includes fields with a text status, an integer code (1=likely unaltered, 2=indeterminate, 3=likely altered), as well as for which direction alteration is (or may be) in if it's indeterminate or likely altered (values are low/high or early/late for timing metrics)

Usage

```
single_metric_alteration(
  metric,
  percentiles,
  predictions,
  ffc_values,
  low_bound_percentile,
  high_bound_percentile,
  prediction_proportion,
  days_in_water_year
)
```

Arguments

`metric`	character name of the metric - case sensitive. Currently only used for timing metrics, which must have "_Tim" in the name
`percentiles`	data frame row - should have a named value "p50" that can be accessed, at the very least. These are the calculated percentile values from the FFC.

`predictions`	data frame (or other named field item) the predicted flow metric values for the segment and metric
`ffc_values`	vector of raw observed metric values (FFC output) for this metric
`low_bound_percentile`	character name of the field in predictions that has the lower bound for normal (default "p10")
`high_bound_percentile`	character name of the field in predictions that has the upper bound for normal (default "p90")
`days_in_water_year`	numeric of how many days in the water year (defaults to 365, but could be 366).
`predicted_proportion`	numeric. When we know that we're not unaltered, we construct an interval to assess if we're altered, which is a two-sided multiplication of the low_bound and the high_bound by (1+prediction_proportion). Typically 0.2 (default "0.2")

`timing_alteration` *So here's a pain in the rear - for timing metrics, none of them have percentiles that cross water years (which seems kind of suspicious to me, but whatever), so the upper bound will never be earlier in the water year than the lower bound - that makes things a bit easier. BUT, for alteration, we have plenty of timing metrics that are predicted to be very early in the water year. If the actual value is early enough that it's in the previous water year (looking at you fall flushing flow), then we don't want to mark it as being *late* when it's actually early! So, we need to have some rules for early and late for timing. Planning to determine the range of values that aren't in the inter-80th percentile range and then find the day of the water year that's in the middle. Timings earlier than that are late, timings after that are early.*

Description

So here's a pain in the rear - for timing metrics, none of them have percentiles that cross water years (which seems kind of suspicious to me, but whatever), so the upper bound will never be earlier in the water year than the lower bound - that makes things a bit easier. BUT, for alteration, we have plenty of timing metrics that are predicted to be very early in the water year. If the actual value is early enough that it's in the previous water year (looking at you fall flushing flow), then we don't want to mark it as being *late* when it's actually early! So, we need to have some rules for early and late for timing. Planning to determine the range of values that aren't in the inter-80th percentile range and then find the day of the water year that's in the middle. Timings earlier than that are late, timings after that are early.

Usage

```
timing_alteration(median_value, low_bound, upper_bound, days_in_water_year)
```

`USGSGage` *USGS Gage Retrieval Tools*

Description

This class retrieves data for a USGS gage.

Details

#library(ffcAPIClient) #gageid <- 11427000 #gage <- USGSGage$new() #gage$id <- 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()
```

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