Package ‘whitewater’

April 1, 2023

Title  Parallel Processing Options for Package 'dataRetrieval'
Version  0.1.3
Description  Provides methods for retrieving United States Geological Survey (USGS) water data using sequential and parallel processing (Bengtsson, 2022 <doi:10.32614/RJ-2021-048>). In addition to parallel methods, data wrangling and additional statistical attributes are provided.


BugReports  https://github.com/joshualerickson/whitewater/issues/
License  MIT + file LICENSE
Encoding  UTF-8
RoxygenNote  7.1.2
Imports  dataRetrieval, dplyr, cli, crayon, furrr, httr, plyr, purrr, stringr, usethis, lubridate, readr, tidyr, future
Suggests  ggplot2, covr, rmarkdown, knitr, ggfx, broom, patchwork, jsonlite, Kendall, testthat (>= 3.0.0)

Config/testthat/edition  3
Depends  R (>= 3.4.0)
LazyData  true
NeedsCompilation  no
Author  Josh Erickson [aut, cre, cph]
Maintainer  Josh Erickson <joshualerickson@gmail.com>
Repository  CRAN
Date/Publication  2023-04-01 18:00:02 UTC

R topics documented:

delay_setup ...................................................... 2
pnw_wy ............................................................ 2
wwOptions .......................................................... 4
### pnw_wy

#### Description

A subset of USGS stations in HUC 17

#### Usage

pnw_wy

---

### delay_setup

<table>
<thead>
<tr>
<th>Description</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay</td>
<td>delay_setup()</td>
</tr>
</tbody>
</table>

#### Value

a number for amount of time to delay
Format

A data frame with 18934 rows and 30 variables:

- **Station**: name of USGS station
- **site_no**: station site id number
- **wy**: water year
- **peak_va**: peak flow value
- **peak_dt**: peak flow date
- **drainage_area**: drainage area in sq.miles
- **lat**: latitude
- **long**: longitude
- **altitude**: altitude in meters
- **obs_per_wy**: observations per water year per site
- **wy_count**: water year count per site
- **Flow_sum**: Sum of Flow
- **Flow_max**: Maximum of Flow
- **Flow_min**: Minimum of Flow
- **Flow_mean**: Mean of Flow
- **Flow_median**: Median of Flow
- **Flow_stdev**: Standard Deviation of Flow
- **Flow_coef_var**: Coeffiecient of Variation of Flow
- **Flow_max_dnorm**: Maximum of Flow normalized by drainage area
- **Flow_min_dnorm**: Minimum of Flow normalized by drainage area
- **Flow_mean_dnorm**: Mean of Flow normalized by drainage area
- **Flow_med_dnorm**: Median of Flow normalized by drainage area
- **Flow_max_sdnorm**: Maximum of Flow normalized by standard deviation
- **Flow_min_sdnorm**: Minimum of Flow normalized by standard deviation
- **Flow_mean_sdnorm**: Mean of Flow normalized by standard deviation
- **Flow_med_sdnorm**: Median of Flow normalized by standard deviation
- **Flow_sd_norm**: Standard Deviation of Flow normalized by standard deviation
- **decade**: decade
- **COMID**: comid of site
- **DamIndex**: dam index

Value

a tibble
Description

Options

Usage

wwOptions(
  date_range = "pfn",
  period = 11,
  dates = NULL,
  site_status = "all",
  floor_iv = "1 hour",
  ...
)

Arguments

date_range A character. Indicating how to call the API. 'pfn' = Period from now, 'date_range' = a date range, "recent" = the most recent value.

period A numeric. Return all values from a period from now (only if 'pfn' is used).

dates A vector. Return all values within an absolute date range (start and end dates). Only if 'date_range' is used.

site_status A character indicating site status. Example, 'all' = both active and inactive, 'active' = only active sites, 'inactive' = only inactive sites.

floor_iv A character on how to floor the instantaneous values, '1 hour' (default).

... other options used for options.

Value

A list with API options.

Note

A site is considered active if; it has collected time-series (automated) data within the last 183 days (6 months) or it has collected discrete (manually collected) data within 397 days (13 months).

Examples

### Not run:

library(whitewater)
yaak_river_dv <- ww_dvUSGS('12304500',
  parameter_cd = '00060',
```r
# wy_month = 10
yaak_river_iv <- ww_floorIVUSGS(yaak_river_dv)

# change floor method
yaak_river_iv <- ww_floorIVUSGS(yaak_river_dv,
    options = wwOptions(floor_iv = '6-hour'))

# change number of days
yaak_river_iv <- ww_floorIVUSGS(yaak_river_dv,
    options = wwOptions(floor_iv = '2-hour',
        period = 365))

# get by date range
yaak_river_wy <- ww_floorIVUSGS(yaak_river_dv,
    options = wwOptions(date_range = 'date_range',
        dates = c('2022-03-01', '2022-05-11')))

# site status as 'active'
yaak_river_wy <- ww_floorIVUSGS(yaak_river_dv,
    options = wwOptions(site_status = 'active',
        date_range = 'date_range',
        dates = c('2022-03-01', '2022-05-11')))

## End(Not run)
```

### Description

Get Current Conditions

### Usage

```r
ww_current_conditions()
```

### Value

A tibble with current conditions and attributes from USGS dashboard.
Note

The time zone used in the URL call is the R session time zone. Also, the time is 1-hour behind. Here are the attributes that are with the data.frame: AgencyCode,SiteNumber,SiteName,SiteTypeCode,Latitude,Longitude, CurrentConditionID,ParameterCode,TimeLocal,TimeZoneCode,Value,ValueFlagCode,RateOfChangeUnitPerHour,StatisticStatusCode,FloodStageStatusCode.

Examples

```r
## Not run:
current_conditions <- ww_current_conditions()

## End(Not run)
```

### ww_dvUSGS

**Process USGS daily values**

**Description**

This function is a wrapper around `readNWISdv` but includes added variables like water year, lat/lon, station name, altitude and tidied dates.

**Usage**

```r
ww_dvUSGS(
sites,
parameter_cd = "00060",
start_date = "",
end_date = "",
stat_cd = "00003",
parallel = FALSE,
wy_month = 10,
verbose = TRUE,
...)
```

**Arguments**

- `sites`: A vector of USGS NWIS sites
- `parameter_cd`: A USGS code for metric, default is "00060".
- `start_date`: A character of date format, e.g. "1990-09-01"
- `end_date`: A character of date format, e.g. "1990-09-01"
- `stat_cd`: character USGS statistic code. This is usually 5 digits. Daily mean (00003) is the default.
- `parallel`: logical indicating whether to use future_map().
Functions: `ww_dvUSGS`, `print.ww_dvUSGS`, `plot.ww_dvUSGS`

**Arguments**

- `huc17_sites`: A data frame with the following columns: `site_no`, `huc`, `siteStatus`, `service`, `parameterCd`, `site_stat`, `siteCd`, `measurementSku`, `measurementCd`, `identifier`. The values represent the site number, basin, site status, service type, parameter code, site status, site code, measurement SKU, measurement code, and identifier, respectively.
- `parameter_cd`: A character string indicating the parameter code, such as '00060' for discharge.
- `wy_month`: A numeric indicating the start month of the water year, e.g., 10 (default).
- `verbose`: A logical for printing information, with TRUE as the default.
- `...`: Additional arguments to pass on to `future_map`

**Value**

A tibble with daily metrics and added meta-data.

**Note**

Use it the same way you would use `readNWISdv`.

**Examples**

```r
library(whitewater)
yaak_river_dv <- ww_dvUSGS('12304500',
                           parameter_cd = '00060',
                           wy_month = 10)
```

```r
#parallel
#get sites
huc17_sites <- dataRetrieval::whatNWISdata(huc = 17,
                                            siteStatus = 'active',
                                            service = 'dv',
                                            parameterCd = '00060')

library(future)
#need to call future::plan()
plan(multisession(workers = availableCores()-1))

pnw_dv <- ww_dvUSGS(huc17_sites$site_no,
                    parameter_cd = '00060',
                    wy_month = 10,
                    parallel = TRUE)
```

**Description**

This function generates instantaneous NWIS data from [https://waterservices.usgs.gov/](https://waterservices.usgs.gov/) and then floors to a user defined interval with `wwOptions` ('1 hour' is default) by taking the mean.
Usage

ww_floorIVUSGS(
  procDV,
  sites = NULL,
  parameter_cd = NULL,
  options = wwOptions(),
  parallel = FALSE,
  verbose = TRUE,
  ...
)

Arguments

procDV A previously created ww_dvUSGS object.
sites A vector of USGS NWIS sites (optional).
parameter_cd A USGS code parameter code, only if using sites argument.
options A wwOptions call.
parallel logical indicating whether to use future_map().
verbose logical for printing information. TRUE (default).
... arguments to pass on to future_map.

Value

A tibble with a user defined interval time step.

Note

For performance reasons, with multi-site retrievals you may retrieve data since October 1, 2007 only. If a previously created ww_dvUSGS object is not used then the user needs to provide a sites vector. This will run ww_dvUSGS in the background.

Examples

## Not run:

library(whitewater)

yaak_river_dv <- ww_dvUSGS('12304500',
 parameter_cd = '00060',
 wy_month = 10)

yaak_river_iv <- ww_floorIVUSGS(yaak_river_dv)

#change floor method

yaak_river_iv <- ww_floorIVUSGS(yaak_river_dv,
 options = wwOptions(floor_iv = '6-hour'))

#change number of days
ww_instantaneousUSGS

Description

This function generates Instantaneous NWIS data from https://waterservices.usgs.gov/.

Usage

```
ww_instantaneousUSGS(
  procDV,
  sites = NULL,
  parameter_cd = NULL,
  options = wwOptions(),
  parallel = FALSE,
)```

```r
yaak_river_iv <- ww_floorIVUSGS(yaak_river_dv,
    options = wwOptions(floor_iv = '2-hour',
                        period = 365))

# get by date range

yaak_river_wy <- ww_floorIVUSGS(yaak_river_dv,
    options = wwOptions(date_range = 'date_range',
                        dates = c('2022-03-01', '2022-05-11')))

# get sites

huc17_sites <- dataRetrieval::whatNWISdata(huc = 17,
                                           siteStatus = 'active',
                                           service = 'dv',
                                           parameterCd = '00060')

library(future)
# need to call future::plan()
plan(multisession(workers = availableCores()-1))

pnw_dv <- ww_dvUSGS(huc17_sites$site_no,
                    parameter_cd = '00060',
                    wy_month = 10,
                    parallel = TRUE)

pnw_iv <- ww_floorIVUSGS(pnw_dv,
                          parallel = TRUE)
```
Arguments

- **procDV**: A previously created `ww_dvUSGS` object.
- **sites**: A vector of USGS NWIS sites. Optional.
- **parameter_cd**: A USGS code parameter code, only if using sites argument.
- **options**: A `wwOptions` call.
- **parallel**: Logical indicating whether to use `future_map()`.
- **verbose**: Logical for printing information. TRUE (default).
- **...**: Arguments to pass on to `future_map`.

Value

A tibble with instantaneous values.

Note

For performance reasons, with multi-site retrievals you may retrieve data since October 1, 2007 only. If a previously created `ww_dvUSGS` object is not used then the user needs to provide a `sites` vector. This will run `ww_dvUSGS` in the background.

Examples

```r
## Not run:
library(whitewater)
yaak_river_dv <- ww_dvUSGS('12304500',
                   parameter_cd = '00060',
                   wy_month = 10)
yaak_river_iv <- ww_instantaneousUSGS(yaak_river_dv)
# change number of days
nyaak_river_iv <- ww_instantaneousUSGS(yaak_river_dv,
                   options = wwOptions(period = 365))
# get by date range
nyaak_river_wy <- ww_instantaneousUSGS(yaak_river_dv,
                   options = wwOptions(date_range = 'date_range',
                                       dates = c('2022-03-01', '2022-05-11')))
# get most recent
nyaak_river_wy <- ww_instantaneousUSGS(yaak_river_dv,
                   options = wwOptions(date_range = 'recent'))
```
# parallel

# get sites

huc17_sites <- dataRetrieval::whatNWISdata(huc = 17,
  siteStatus = 'active',
  service = 'dv',
  parameterCd = '00060')

library(future)
# need to call future::plan()
plan(multiprocess(workers = availableCores()-1))

pnw_dv <- ww_dvUSGS(huc17_sites$site_no,
  parameter_cd = '00060',
  wy_month = 10,
  parallel = TRUE)

pnw_iv <- ww_instantaneousUSGS(pnw_dv,
  parallel = TRUE)

## End(Not run)

---

**ww_monthUSGS**  
*Month-Only Stats (USGS)*

**Description**

This function uses the results of the `ww_dvUSGS` object to generate mean, maximum, median, standard deviation and coefficient of variation for month only.

**Usage**

`ww_monthUSGS(procDV, sites = NULL, parallel = FALSE, verbose = TRUE, ...)`

**Arguments**

- **procDV**  
  A previously created `ww_dvUSGS` object.

- **sites**  
  A character vector with NWIS site numbers (optional).

- **parallel**  
  Logical indicating whether to use `future_map()`.

- **verbose**  
  Logical for printing information. TRUE (default).

- **...**  
  Arguments to pass on to `future_map` and `ww_dvUSGS`.

**Value**

A tibble filtered by month and added meta-data.
Note

If a previously created \texttt{ww\_dvUSGS} object is not used then the user needs to provide a \texttt{sites} vector. This will run \texttt{ww\_dvUSGS} in the background.

Examples

\begin{verbatim}
## Not run:
library(whitewater)
yaak_river_dw <- ww_dvUSGS('12304500',
parameter_cd = '00060',
wy_month = 10)

yaak_river_month <- ww_monthUSGS(yaak_river_dw)

## End(Not run)
\end{verbatim}

\textbf{ww\_peakUSGS} \hspace{1cm} \emph{Get Peak Flows}

\textbf{Description}

Get Peak Flows

\textbf{Usage}

\texttt{ww\_peakUSGS(sites, parallel = FALSE, wy\_month = 10, verbose = TRUE, ...)}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{sites} \hspace{1cm} A vector of USGS NWIS sites
  \item \texttt{parallel} \hspace{1cm} logical indicating whether to use future\_map()
  \item \texttt{wy\_month} \hspace{1cm} numeric indicating the start month of the water year. e.g. 10 (default).
  \item \texttt{verbose} \hspace{1cm} logical for printing information. TRUE (default).
  \item \texttt{...} \hspace{1cm} arguments to pass on to future\_map.
\end{itemize}

\textbf{Value}

\texttt{a tibble} with peaks by water year
Description

This function uses the readNWISstat to gather daily, monthly or yearly percentiles.

Usage

ww_statsUSGS(
  procDV,
  sites = NULL,
  temporalFilter = "daily",
  parameter_cd = NULL,
  days = 10,
  parallel = FALSE,
  verbose = TRUE,
  ...
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>procDV</td>
<td>A previously created ww_dvUSGS object.</td>
</tr>
<tr>
<td>sites</td>
<td>A character USGS NWIS site.</td>
</tr>
<tr>
<td>temporalFilter</td>
<td>A character for the stat summary window, e.g. 'daily' (default), 'monthly', 'yearly'.</td>
</tr>
<tr>
<td>parameter_cd</td>
<td>A USGS code parameter code, only if using sites argument.</td>
</tr>
<tr>
<td>days</td>
<td>A numeric input of days to go back from today (only needed if using .temporalFilter = 'daily').</td>
</tr>
<tr>
<td>parallel</td>
<td>logical indicating whether to use future_map().</td>
</tr>
<tr>
<td>verbose</td>
<td>logical for printing information. TRUE (default).</td>
</tr>
<tr>
<td>...</td>
<td>arguments to pass on to future_map.</td>
</tr>
</tbody>
</table>

Value

a tibble with associated site statistics.

Note

Be aware, the parameter values ('Flow', 'Wtemp', etc) are calculated from the ww_floorIVUSGS function by taking the daily mean of the hourly data. Thus, the instantaneous values will look different than the daily mean values, as it should. The .temporalFilter argument is used to generate the window of percentiles.
Examples

```r
## Not run:
# get by date range

yaak_river_dv <- ww_dvUSGS('12304500')

daily
yaak_river_stats <- ww_statsUSGS(yaak_river_dv,
    temporalFilter = 'daily',
    days = 10)

monthly
yaak_river_stats <- ww_statsUSGS(yaak_river_dv,
    temporalFilter = 'monthly',
    days = 10)

yearly
yaak_river_stats <- ww_statsUSGS(yaak_river_dv,
    temporalFilter = 'yearly',
    days = 10)

## End(Not run)
```

---

**ww_wymUSGS**

*Water Year & Monthly Stats (USGS)*

**Description**

This function uses the results of the `ww_dvUSGS` object to generate mean, maximum, median, standard deviation and coefficient of variation per water year per month.

**Usage**

`ww_wymUSGS(procDV, sites = NULL, parallel = FALSE, verbose = TRUE, ...)`

**Arguments**

- `procDV` A previously created `ww_dvUSGS` object.
- `sites` A character vector with NWIS site numbers (optional).
- `parallel` logical indicating whether to use `future_map()`.
- `verbose` logical for printing information. TRUE (default).
- `...` arguments to pass on to `future_map` and `ww_dvUSGS`.

**Value**

A tibble filtered by water year and month with added meta-data.

**Note**

If a previously created `ww_dvUSGS` object is not used then the user needs to provide a `sites` vector. This will run `ww_dvUSGS` in the background.
## Not run:

```r
library(whitewater)
yaak_river_dv <- ww_dvUSGS('12304500',
parameter_cd = '00060',
wy_month = 10)

yaak_river_wym <- ww_wymUSGS(yaak_river_dv)
```

## End(Not run)

### Description

This function uses the results of the `ww_dvUSGS` object to generate mean, maximum, median, standard deviation and some normalization methods (drainage area, scaled by log and standard deviation) per water year.

### Usage

```r
ww_wyUSGS(procDV, sites = NULL, parallel = FALSE, verbose = TRUE, ...)
```

### Arguments

- `procDV` A previously created `ww_dvUSGS` object.
- `sites` A character vector with NWIS site numbers (optional).
- `parallel` logical indicating whether to use `future_map()`.
- `verbose` logical for printing information. TRUE (default).
- `...` arguments to pass on to `future_map` and/or `ww_dvUSGS`.

### Value

A tibble filtered by water year with added meta-data.

### Note

If a previously created `ww_dvUSGS` object is not used then the user needs to provide a `sites` vector. This will run `ww_dvUSGS` in the background.
Examples

## Not run:

```r
library(whitewater)
yaak_river_dv <- ww_dvUSGS('
12304500',
parameter_cd = '00060',
wmonth = 10)
yaak_river_wy <- ww_wyUSGS(yaak_river_dv)

# parallel

# get sites

huc17_sites <- dataRetrieval::whatNWISdata(huc = 17,
siteStatus = 'active',
service = 'dv',
parameterCd = '00060')

library(future)
# need to call future::plan()
plan(multisession(workers = availableCores()-1))

pnw_dv <- ww_dvUSGS(huc17_sites$site_no,
parameter_cd = '00060',
wmonth = 10,
parallel = TRUE)

pnw_wy <- ww_wyUSGS(pnw_dv,
parallel = TRUE)

## End(Not run)
```
Index

* datasets
  pnw_wy, 2

delay_setup, 2

future_map, 7, 8, 10–15

pnw_wy, 2

readNWISdv, 6, 7
readNWISstat, 13

ww_current_conditions, 5
ww_dvUSGS, 6, 8, 10–15
ww_floorIVUSGS, 7, 13
ww_instantaneousUSGS, 9
ww_monthUSGS, 11
ww_peakUSGS, 12
ww_statsUSGS, 13
ww_wymUSGS, 14
ww_wyUSGS, 15
wwOptions, 4, 7, 8, 10