Package: nwmTools (via r-universe)

December 4, 2024

Type Package

Title nwmTools

Description Tools for working with operational and historic National Water Model Output.

Version 0.0.4

Maintainer Mike Johnson <mike.johnson@noaa.gov>

URL https://github.com/mikejohnson51/nwmTools

BugReports https://github.com/mikejohnson51/nwmTools/issues

Depends R(>= 3.5.0)

Imports dataRetrieval, dplyr, glue, httr, lubridate, nhdplusTools, RNetCDF, rvest, terra, xml2, utils

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Suggests AOI, testthat (>= 3.0.0)

Remotes mikejohnson51/AOI

Config/testthat/edition 3

License CC0

Config/pak/sysreqs cmake libgdal-dev gdal-bin libgeos-dev libicu-dev libpng-dev libxml2-dev libnetcdf-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev libx11-dev

Repository https://mikejohnson51.r-universe.dev

RemoteUrl https://github.com/mikejohnson51/nwmTools

RemoteRef HEAD

RemoteSha 5005f4e5f56d8cdbbe8fd612a27c278c176a2341

Contents

`add_waterYear` *Add Water Year Column*

Description

Add Water Year Column

Usage

```
add_waterYear(dateVec)
```

Arguments

`dateVec`	raw data returned from readNWMdata

Value

vector of water years

`aggregate_dowy` *Aggregate by DOWY*

Description

Aggregate by DOWY

Usage

```
aggregate_dowy(rawData, fun = "mean", na.rm = TRUE)
```

Arguments

`rawData`	data extracted with `readNWMdata`
`fun`	function to be applied to the flows column default = 'mean'
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds.

Details

NWM data is extracted as hourly values.

To aggregate hourly data to different time chunks the nwmHistoric package offers a family of aggregate functions.

Each of these begins with the prefix 'aggregate_' and is followed by the date symbol to aggregate to.

Symbol	Aggregate
y	year
m	month
d	day
j	julien day
s	season
wy	water year

See Also

Other aggregate functions: `aggregate_j()`, `aggregate_m()`, `aggregate_record()`, `aggregate_s()`, `aggregate_wymd()`, `aggregate_wym()`, `aggregate_wys()`, `aggregate_wy()`, `aggregate_yj()`, `aggregate_ymd()`, `aggregate_ym()`, `aggregate_ys()`, `aggregate_y()`

Examples

```
## Not run:
# Get flow record for COMID 101
flows = readNWMdata(comid = 101)

# Aggregate to yearly average (y)
yearly = aggregate_y(flows, fun = 'mean')

# Aggregate to monthly
# minimum and maximum per year (ym)
ym = aggregate_ym(flows, fun = list(min, max))

# Aggregate to seasonal 95th percetile
# with using custom function
s95 = aggregate_s(flows, fun = function(x){quantile(x,.95)})

## End(Not run)
```

`aggregate_j` *Aggregate by Julien Day*

Description

Aggregate by Julien Day

Usage

```
aggregate_j(rawData, fun = "mean", na.rm = TRUE)
```

Arguments

`rawData`	data extracted with `readNWMdata`
`fun`	function to be applied to the flows column default = 'mean'
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds.

Details

NWM data is extracted as hourly values.

To aggregate hourly data to different time chunks the nwmHistoric package offers a family of aggregate functions.

Each of these begins with the prefix 'aggregate_' and is followed by the date symbol to aggregate to.

Symbol	Aggregate
y	year
m	month

d	day
j	julien day
s	season
wy	water year

See Also

Other aggregate functions: `aggregate_dowy()`, `aggregate_m()`, `aggregate_record()`, `aggregate_s()`, `aggregate_wymd()`, `aggregate_wym()`, `aggregate_wys()`, `aggregate_wy()`, `aggregate_yj()`, `aggregate_ymd()`, `aggregate_ym()`, `aggregate_ys()`, `aggregate_y()`

Examples

```
## Not run:
# Get flow record for COMID 101
flows = readNWMdata(comid = 101)

# Aggregate to yearly average (y)
yearly = aggregate_y(flows, fun = 'mean')

# Aggregate to monthly
# minimum and maximum per year (ym)
ym = aggregate_ym(flows, fun = list(min, max))

# Aggregate to seasonal 95th percetile
# with using custom function
s95 = aggregate_s(flows, fun = function(x){quantile(x,.95)})

## End(Not run)
```

---

aggregate_m *Aggregate by Month*

Description

Aggregate by Month

Usage

```
aggregate_m(rawData, fun = "mean", na.rm = TRUE)
```

Arguments

`rawData`	data extracted with `readNWMdata`
`fun`	function to be applied to the flows column default = 'mean'
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds.

Details

NWM data is extracted as hourly values.

To aggregate hourly data to different time chunks the nwmHistoric package offers a family of aggregate functions.

Each of these begins with the prefix 'aggregate_' and is followed by the date symbol to aggregate to.

Symbol	Aggregate
y	year
m	month
d	day
j	julien day
s	season
wy	water year

See Also

Other aggregate functions: `aggregate_dowy()`, `aggregate_j()`, `aggregate_record()`, `aggregate_s()`, `aggregate_wymd()`, `aggregate_wym()`, `aggregate_wys()`, `aggregate_wy()`, `aggregate_yj()`, `aggregate_ymd()`, `aggregate_ym()`, `aggregate_ys()`, `aggregate_y()`

Examples

```
## Not run:
# Get flow record for COMID 101
flows = readNWMdata(comid = 101)

# Aggregate to yearly average (y)
yearly = aggregate_y(flows, fun = 'mean')

# Aggregate to monthly
# minimum and maximum per year (ym)
ym = aggregate_ym(flows, fun = list(min, max))

# Aggregate to seasonal 95th percetile
# with using custom function
s95 = aggregate_s(flows, fun = function(x){quantile(x,.95)})

## End(Not run)
```

`aggregate_record` *Aggregate by Record*

Description

Aggregate by Record

Usage

```
aggregate_record(rawData, fun = "mean", na.rm = TRUE)
```

Arguments

`rawData`	data extracted with `readNWMdata`
`fun`	function to be applied to the flows column default = 'mean'
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds.

Details

NWM data is extracted as hourly values.

To aggregate hourly data to different time chunks the nwmHistoric package offers a family of aggregate functions.

Each of these begins with the prefix 'aggregate_' and is followed by the date symbol to aggregate to.

Symbol	Aggregate
y	year
m	month
d	day
j	julien day
s	season
wy	water year

See Also

Other aggregate functions: `aggregate_dowy()`, `aggregate_j()`, `aggregate_m()`, `aggregate_s()`, `aggregate_wymd()`, `aggregate_wym()`, `aggregate_wys()`, `aggregate_wy()`, `aggregate_yj()`, `aggregate_ymd()`, `aggregate_ym()`, `aggregate_ys()`, `aggregate_y()`

Examples

```
## Not run:
# Get flow record for COMID 101
flows = readNWMdata(comid = 101)

# Aggregate to yearly average (y)
yearly = aggregate_y(flows, fun = 'mean')

# Aggregate to monthly
# minimum and maximum per year (ym)
ym = aggregate_ym(flows, fun = list(min, max))

# Aggregate to seasonal 95th percetile
# with using custom function
s95 = aggregate_s(flows, fun = function(x){quantile(x,.95)})
```

```
## End(Not run)
```

aggregate_s *Aggregate by Season*

Description

Aggregate by Season

Usage

```
aggregate_s(rawData, fun = "mean", na.rm = TRUE)
```

Arguments

`rawData`	data extracted with `readNWMdata`
`fun`	function to be applied to the flows column default = 'mean'
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds.

Details

NWM data is extracted as hourly values.

To aggregate hourly data to different time chunks the nwmHistoric package offers a family of aggregate functions.

Each of these begins with the prefix 'aggregate_' and is followed by the date symbol to aggregate to.

Symbol	Aggregate
y	year
m	month
d	day
j	julien day
s	season
wy	water year

See Also

Other aggregate functions: `aggregate_dowy()`, `aggregate_j()`, `aggregate_m()`, `aggregate_record()`, `aggregate_wymd()`, `aggregate_wym()`, `aggregate_wys()`, `aggregate_wy()`, `aggregate_yj()`, `aggregate_ymd()`, `aggregate_ym()`, `aggregate_ys()`, `aggregate_y()`

Examples

```
## Not run:
# Get flow record for COMID 101
flows = readNWMdata(comid = 101)

# Aggregate to yearly average (y)
yearly = aggregate_y(flows, fun = 'mean')

# Aggregate to monthly
# minimum and maximum per year (ym)
ym = aggregate_ym(flows, fun = list(min, max))

# Aggregate to seasonal 95th percetile
# with using custom function
s95 = aggregate_s(flows, fun = function(x){quantile(x,.95)})

## End(Not run)
```

---

aggregate_wy *Aggregate by Water Year*

Description

Aggregate by Water Year

Usage

```
aggregate_wy(rawData, fun = "mean", na.rm = TRUE)
```

Arguments

`rawData`	data extracted with `readNWMdata`
`fun`	function to be applied to the flows column default = 'mean'
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds.

Details

NWM data is extracted as hourly values.

To aggregate hourly data to different time chunks the nwmHistoric package offers a family of aggregate functions.

Each of these begins with the prefix 'aggregate_' and is followed by the date symbol to aggregate to.

Symbol	Aggregate
y	year
m	month

d	day
j	julien day
s	season
wy	water year

See Also

Other aggregate functions: aggregate_dowy(), aggregate_j(), aggregate_m(), aggregate_record(), aggregate_s(), aggregate_wymd(), aggregate_wym(), aggregate_wys(), aggregate_yj(), aggregate_ymd(), aggregate_ym(), aggregate_ys(), aggregate_y()

Examples

```
## Not run:
# Get flow record for COMID 101
flows = readNWMdata(comid = 101)

# Aggregate to yearly average (y)
yearly = aggregate_y(flows, fun = 'mean')

# Aggregate to monthly
# minimum and maximum per year (ym)
ym = aggregate_ym(flows, fun = list(min, max))

# Aggregate to seasonal 95th percetile
# with using custom function
s95 = aggregate_s(flows, fun = function(x){quantile(x,.95)})

## End(Not run)
```

---

aggregate_wym *Aggregate by Water Year - Month*

Description

Aggregate by Water Year - Month

Usage

```
aggregate_wym(rawData, fun = "mean", na.rm = TRUE)
```

Arguments

rawData	data extracted with readNWMdata
fun	function to be applied to the flows column default = 'mean'
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.

Details

NWM data is extracted as hourly values.

To aggregate hourly data to different time chunks the nwmHistoric package offers a family of aggregate functions.

Each of these begins with the prefix 'aggregate_' and is followed by the date symbol to aggregate to.

Symbol	Aggregate
y	year
m	month
d	day
j	julien day
s	season
wy	water year

See Also

Other aggregate functions: `aggregate_dowy()`, `aggregate_j()`, `aggregate_m()`, `aggregate_record()`, `aggregate_s()`, `aggregate_wymd()`, `aggregate_wys()`, `aggregate_wy()`, `aggregate_yj()`, `aggregate_ymd()`, `aggregate_ym()`, `aggregate_ys()`, `aggregate_y()`

Examples

```
## Not run:
# Get flow record for COMID 101
flows = readNWMdata(comid = 101)

# Aggregate to yearly average (y)
yearly = aggregate_y(flows, fun = 'mean')

# Aggregate to monthly
# minimum and maximum per year (ym)
ym = aggregate_ym(flows, fun = list(min, max))

# Aggregate to seasonal 95th percetile
# with using custom function
s95 = aggregate_s(flows, fun = function(x){quantile(x,.95)})

## End(Not run)
```

`aggregate_wymd` *Aggregate by Water Year - Month - Day*

Description

Aggregate by Water Year - Month - Day

Usage

```
aggregate_wymd(rawData, fun = "mean", na.rm = TRUE)
```

Arguments

`rawData`	data extracted with `readNWMdata`
`fun`	function to be applied to the flows column default = 'mean'
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds.

Details

NWM data is extracted as hourly values.

To aggregate hourly data to different time chunks the nwmHistoric package offers a family of aggregate functions.

Each of these begins with the prefix 'aggregate_' and is followed by the date symbol to aggregate to.

Symbol	Aggregate
y	year
m	month
d	day
j	julien day
s	season
wy	water year

See Also

Other aggregate functions: `aggregate_dowy()`, `aggregate_j()`, `aggregate_m()`, `aggregate_record()`, `aggregate_s()`, `aggregate_wym()`, `aggregate_wys()`, `aggregate_wy()`, `aggregate_yj()`, `aggregate_ymd()`, `aggregate_ym()`, `aggregate_ys()`, `aggregate_y()`

Examples

```
## Not run:
# Get flow record for COMID 101
flows = readNWMdata(comid = 101)

# Aggregate to yearly average (y)
yearly = aggregate_y(flows, fun = 'mean')

# Aggregate to monthly
# minimum and maximum per year (ym)
ym = aggregate_ym(flows, fun = list(min, max))

# Aggregate to seasonal 95th percetile
# with using custom function
s95 = aggregate_s(flows, fun = function(x){quantile(x,.95)})
```

```
## End(Not run)
```

`aggregate_wys` *Aggregate by Water Year - Season*

Description

Aggregate by Water Year - Season

Usage

```
aggregate_wys(rawData, fun = "mean", na.rm = TRUE)
```

Arguments

`rawData`	data extracted with `readNWMdata`
`fun`	function to be applied to the flows column default = 'mean'
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds.

Details

NWM data is extracted as hourly values.

To aggregate hourly data to different time chunks the nwmHistoric package offers a family of aggregate functions.

Each of these begins with the prefix 'aggregate_' and is followed by the date symbol to aggregate to.

Symbol	Aggregate
y	year
m	month
d	day
j	julien day
s	season
wy	water year

See Also

Other aggregate functions: `aggregate_dowy()`, `aggregate_j()`, `aggregate_m()`, `aggregate_record()`, `aggregate_s()`, `aggregate_wymd()`, `aggregate_wym()`, `aggregate_wy()`, `aggregate_yj()`, `aggregate_ymd()`, `aggregate_ym()`, `aggregate_ys()`, `aggregate_y()`

Examples

```
## Not run:
# Get flow record for COMID 101
flows = readNWMdata(comid = 101)

# Aggregate to yearly average (y)
yearly = aggregate_y(flows, fun = 'mean')

# Aggregate to monthly
# minimum and maximum per year (ym)
ym = aggregate_ym(flows, fun = list(min, max))

# Aggregate to seasonal 95th percetile
# with using custom function
s95 = aggregate_s(flows, fun = function(x){quantile(x,.95)})

## End(Not run)
```

---

aggregate_y *Aggregate by Year*

Description

Aggregate by Year

Usage

```
aggregate_y(rawData, fun = "mean", na.rm = TRUE)
```

Arguments

`rawData`	data extracted with `readNWMdata`
`fun`	function to be applied to the flows column default = 'mean'
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds.

Details

NWM data is extracted as hourly values.

To aggregate hourly data to different time chunks the nwmHistoric package offers a family of aggregate functions.

Each of these begins with the prefix 'aggregate_' and is followed by the date symbol to aggregate to.

Symbol	Aggregate
y	year
m	month

d	day
j	julien day
s	season
wy	water year

See Also

Other aggregate functions: `aggregate_dowy()`, `aggregate_j()`, `aggregate_m()`, `aggregate_record()`, `aggregate_s()`, `aggregate_wymd()`, `aggregate_wym()`, `aggregate_wys()`, `aggregate_wy()`, `aggregate_yj()`, `aggregate_ymd()`, `aggregate_ym()`, `aggregate_ys()`

Examples

```
## Not run:
# Get flow record for COMID 101
flows = readNWMdata(comid = 101)

# Aggregate to yearly average (y)
yearly = aggregate_y(flows, fun = 'mean')

# Aggregate to monthly
# minimum and maximum per year (ym)
ym = aggregate_ym(flows, fun = list(min, max))

# Aggregate to seasonal 95th percetile
# with using custom function
s95 = aggregate_s(flows, fun = function(x){quantile(x,.95)})

## End(Not run)
```

`aggregate_yj` *Aggregate by Year-Julien Day*

Description

Aggregate by Year-Julien Day

Usage

```
aggregate_yj(rawData, fun = "mean", na.rm = TRUE)
```

Arguments

`rawData`	data extracted with `readNWMdata`
`fun`	function to be applied to the flows column default = 'mean'
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds.

Details

NWM data is extracted as hourly values.

To aggregate hourly data to different time chunks the nwmHistoric package offers a family of aggregate functions.

Each of these begins with the prefix 'aggregate_' and is followed by the date symbol to aggregate to.

Symbol	Aggregate
y	year
m	month
d	day
j	julien day
s	season
wy	water year

See Also

Other aggregate functions: `aggregate_dowy()`, `aggregate_j()`, `aggregate_m()`, `aggregate_record()`, `aggregate_s()`, `aggregate_wymd()`, `aggregate_wym()`, `aggregate_wys()`, `aggregate_wy()`, `aggregate_ymd()`, `aggregate_ym()`, `aggregate_ys()`, `aggregate_y()`

Examples

```
## Not run:
# Get flow record for COMID 101
flows = readNWMdata(comid = 101)

# Aggregate to yearly average (y)
yearly = aggregate_y(flows, fun = 'mean')

# Aggregate to monthly
# minimum and maximum per year (ym)
ym = aggregate_ym(flows, fun = list(min, max))

# Aggregate to seasonal 95th percetile
# with using custom function
s95 = aggregate_s(flows, fun = function(x){quantile(x,.95)})

## End(Not run)
```

`aggregate_ym` *Aggregate by Year-Month*

Description

Aggregate by Year-Month

Usage

```
aggregate_ym(rawData, fun = "mean", na.rm = TRUE)
```

Arguments

`rawData`	data extracted with `readNWMdata`
`fun`	function to be applied to the flows column default = 'mean'
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds.

Details

NWM data is extracted as hourly values.

To aggregate hourly data to different time chunks the nwmHistoric package offers a family of aggregate functions.

Each of these begins with the prefix 'aggregate_' and is followed by the date symbol to aggregate to.

Symbol	Aggregate
y	year
m	month
d	day
j	julien day
s	season
wy	water year

See Also

Other aggregate functions: `aggregate_dowy()`, `aggregate_j()`, `aggregate_m()`, `aggregate_record()`, `aggregate_s()`, `aggregate_wymd()`, `aggregate_wym()`, `aggregate_wys()`, `aggregate_wy()`, `aggregate_yj()`, `aggregate_ymd()`, `aggregate_ys()`, `aggregate_y()`

Examples

```
## Not run:
# Get flow record for COMID 101
flows = readNWMdata(comid = 101)

# Aggregate to yearly average (y)
yearly = aggregate_y(flows, fun = 'mean')

# Aggregate to monthly
# minimum and maximum per year (ym)
ym = aggregate_ym(flows, fun = list(min, max))

# Aggregate to seasonal 95th percetile
# with using custom function
s95 = aggregate_s(flows, fun = function(x){quantile(x,.95)})
```

```
## End(Not run)
```

---

aggregate_ymd *Aggregate by Year-Month-Day*

Description

Aggregate by Year-Month-Day

Usage

```
aggregate_ymd(rawData, fun = "mean", na.rm = TRUE)
```

Arguments

`rawData`	data extracted with `readNWMdata`
`fun`	function to be applied to the flows column default = 'mean'
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds.

Details

NWM data is extracted as hourly values.

To aggregate hourly data to different time chunks the nwmHistoric package offers a family of aggregate functions.

Each of these begins with the prefix 'aggregate_' and is followed by the date symbol to aggregate to.

Symbol	Aggregate
y	year
m	month
d	day
j	julien day
s	season
wy	water year

See Also

Other aggregate functions: `aggregate_dowy()`, `aggregate_j()`, `aggregate_m()`, `aggregate_record()`, `aggregate_s()`, `aggregate_wymd()`, `aggregate_wym()`, `aggregate_wys()`, `aggregate_wy()`, `aggregate_yj()`, `aggregate_ym()`, `aggregate_ys()`, `aggregate_y()`

Examples

```
## Not run:
# Get flow record for COMID 101
flows = readNWMdata(comid = 101)

# Aggregate to yearly average (y)
yearly = aggregate_y(flows, fun = 'mean')

# Aggregate to monthly
# minimum and maximum per year (ym)
ym = aggregate_ym(flows, fun = list(min, max))

# Aggregate to seasonal 95th percetile
# with using custom function
s95 = aggregate_s(flows, fun = function(x){quantile(x,.95)})

## End(Not run)
```

`aggregate_ys` *Aggregate by Year-Season*

Description

Aggregate by Year-Season

Usage

```
aggregate_ys(rawData, fun = "mean", na.rm = TRUE)
```

Arguments

`rawData`	data extracted with `readNWMdata`
`fun`	function to be applied to the flows column default = 'mean'
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds.

Details

NWM data is extracted as hourly values.

To aggregate hourly data to different time chunks the nwmHistoric package offers a family of aggregate functions.

Each of these begins with the prefix 'aggregate_' and is followed by the date symbol to aggregate to.

Symbol	Aggregate
y	year
m	month

d	day
j	julien day
s	season
wy	water year

See Also

Other aggregate functions: `aggregate_dowy()`, `aggregate_j()`, `aggregate_m()`, `aggregate_record()`, `aggregate_s()`, `aggregate_wymd()`, `aggregate_wym()`, `aggregate_wys()`, `aggregate_wy()`, `aggregate_yj()`, `aggregate_ymd()`, `aggregate_ym()`, `aggregate_y()`

Examples

```
## Not run:
# Get flow record for COMID 101
flows = readNWMdata(comid = 101)

# Aggregate to yearly average (y)
yearly = aggregate_y(flows, fun = 'mean')

# Aggregate to monthly
# minimum and maximum per year (ym)
ym = aggregate_ym(flows, fun = list(min, max))

# Aggregate to seasonal 95th percetile
# with using custom function
s95 = aggregate_s(flows, fun = function(x){quantile(x,.95)})

## End(Not run)
```

`crop_flipped_nwm` *Crop Flipped Raster*

Description

Crop Flipped Raster

Usage

```
crop_flipped_nwm(x, AOI)
```

Arguments

x	SpatRast object
AOI	a sf polygon

Value

SpatRast object (x cropped to AOI)

`download_files` *Download Remote Files*

Description

Download Remote Files

Usage

```
download_files(fileList, outdir = ".")
```

Arguments

`fileList`	fileList object
`outdir`	directory to write files

Value

data.frame

`get_aws_urls` *Get GCP file list*

Description

Get GCP file list

Usage

```
get_aws_urls(
  version = 2.1,
  output = "CHRTOUT",
  config = NULL,
  ensemble = NULL,
  date = "2010-10-29",
  hour = "00",
  minute = "00",
  num = 3,
  outdir = NULL
)
```

Arguments

`version`	NWM model version
`output`	the NWM model output type
`config`	the NWM model configurarion
`ensemble`	the NWM ensemble number
`date`	date of interest
`hour`	hour of interest
`minute`	minute of interest
`num`	number of files to get (forward from provides data-hour-minute)

Value

data.frame

`get_gcp_urls` *Get GCP file list*

Description

Get GCP file list

Usage

```
get_gcp_urls(
  config = "short_range",
  domain = "conus",
  date,
  hour = "00",
  minute = "00",
  num,
  ensemble = NULL,
  output = "channel_rt"
)
```

Arguments

`config`	the NWM model configurarion
`domain`	the NWM model domain
`date`	date of interest
`hour`	hour of interest
`minute`	minute of interest
`num`	number of files to get (forward from provides data-hour-minute)
`ensemble`	the NWM ensemble number
`output`	the NWM model output type

Value

data.frame

`get_gridded_data` *Extract Gridded Data from fileList*

Description

Extract Gridded Data from fileList

Usage

```
get_gridded_data(fileList, AOI, varname, outfile = NULL)
```

Arguments

`fileList`	a list of gridded NWM outputs
`AOI`	area of interest (sf POLYGON) to subset
`varname`	the name of the variable to extract
`outfile`	filepath to save data (with .nc extension)

Value

data.frame

`get_gridded_local` *Extract Gridded Data from local fileList*

Description

Extract Gridded Data from local fileList

Usage

```
get_gridded_local(fileList, AOI, varname, outfile = NULL)
```

Arguments

`fileList`	a list of gridded NWM outputs
`AOI`	area of interest (sf POLYGON) to subset
`varname`	the name of the variable to extract
`outfile`	filepath to save data (with .nc extension)

Value

data.frame

get_nomads_urls *Get NOMADs File List*

Description

Get NOMADs File List

Usage

```
get_nomads_urls(
  config = "short_range",
  domain = "conus",
  date = NULL,
  hour = NULL,
  minute = "00",
  num,
  ensemble = NULL,
  output = "channel_rt",
  version = "prod",
  outdir = NULL
)
```

Arguments

config	the NWM model configurarion
domain	the NWM model domain
date	date of interest
hour	hour of interest
minute	minute of interest
num	number of files to get (forward from provides data-hour-minute)
ensemble	the NWM ensemble number
output	the NWM model output type
version	server version (prod or para)

Value

data.frame

`get_nwm_meta` *NWM metadata*

Description

NWM metadata

Usage

```
get_nwm_meta(version = NULL)
```

Arguments

`version`	Which version of NWM should be returned? (1.2, 2.0, 2.1)

Value

data.frame

`get_timeseries` *Extract Timeseries from file list*

Description

Extract Timeseries from file list

Usage

```
get_timeseries(
  fileList,
  ids = NULL,
  index_id = "feature_id",
  varname = "streamflow",
  outfile = NULL
)
```

Arguments

`fileList`	a list of non-gridded NWM outputs
`ids`	a set of ids to limit the returned data to
`index_id`	the name of the id attributes
`varname`	the name of the variable
`outfile`	file path to save data to (.nc extension)

Value

data.frame

`get_timeseries_local` *Extract Timeseries from local file list*

Description

Extract Timeseries from local file list

Usage

```
get_timeseries_local(
  fileList,
  ids = NULL,
  index_id = "feature_id",
  varname = "streamflow",
  outfile = NULL
)
```

Arguments

`fileList`	a list of non-gridded NWM outputs
`ids`	a set of ids to limit the returned data to
`index_id`	the name of the id attributes
`varname`	the name of the variable
`outfile`	file path to save data to (.nc extension)

Value

data.frame

`nwm_data` *NWM Data Types*

Description

NWM Data Types

Usage

```
nwm_data
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 603 rows and 13 columns.

readNWMdata *NWM Reanalysis Extraction*

Description

Download hourly flow values for an NHD COMID from the National Water Model version 1.2 or 2.0. Returned data is available between "1993-01-01 00" and "2017-12-31 23" but can be subset using a startDate and endDate.

Usage

```
readNWMdata(
  AOI = NULL,
  comid = NULL,
  siteID = NULL,
  startDate = NULL,
  endDate = NULL,
  tz = "UTC",
  version = 2.1,
  addObs = FALSE,
  add_nhd = FALSE
)
```

Arguments

`AOI`	spatial polygon or point to extract data for
`comid`	a NHD common identifier
`siteID`	a USGS NWIS site identifier (eight digits)
`startDate`	a start date (YYYY-MM-DD) or (YYYY-MM-DD HH)
`endDate`	an end date (YYYY-MM-DD) or (YYYY-MM-DD HH)
`tz`	the desired timezone of the data. Can be found with `OlsonNames`
`version`	the NWM version to extract (current = 1.2 or 2 (default))
`addObs`	should observation data be added? Only available when !is.null(siteID)
`add_nhd`	should the NHD spatial features be added to the output

Value

data.frame or sf object

Examples

```
## Not run:
readNWMdata(comid = 101)
readNWMdata(comid = 101, version = 1.2)
readNWMdata(comid = 101, tz = "US/Pacific")

## End(Not run)
```

`split_time` *Split Y-M-D-H into time components*

Description

Split Y-M-D-H into time components

Usage

```
split_time(rawData, time_col)
```

Arguments

`rawData`	rawData with time column
`time_col`	the column name holding dateTime

Value

data.frame with added time components

`tds_meta` *NWM THREDDS Servers*

Description

NWM THREDDS Servers

Usage

```
tds_meta
```

Format

An object of class `data.frame` with 3 rows and 7 columns.

Index