Package 'AWAPer'

August 14, 2020

Type Package

Title Catchment Area Weighted Climate Data Anywhere in Australia

Version 0.1.42

Maintainer Tim Peterson <tim.peterson@monash.edu>

Description NetCDF files of the Bureau of Meteorology Australian Water Availability Project daily national climate grids are built and used for the efficient extraction of point and catchment area weighted precipitation, minimum temperature, maximum temperature, vapour pressure, solar radiation and various measures of evapotranspiration. For details on the source climate data see <http://www.bom.gov.au/jsp/awap/>.

Depends R (>= 3.2.3)

Imports Evapotranspiration (>= 1.14), ncdf4, R.utils, raster, chron, maptools, sp, zoo, methods, xts

BugReports https://github.com/peterson-tim-j/AWAPer/issues

URL https://github.com/peterson-tim-j/AWAPer

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

SystemRequirements 7z (Windows only)

Author Tim Peterson [aut, cre], Conrad Wasko [ctb]

R topics documented:

catchments *Example catchment boundary polygons.*

Description

Two example catchment boundaries as a SpatialPolygonsDataFrame. The catchments are Creswick Creek (ID 407214, Vic., Australia, see http://www.bom.gov.au/water/hrs/#id=407214) and Bet Bet Creek (ID 407220, Vic., Australia, see http://www.bom.gov.au/water/hrs/#id=407220). The catchments can be used to extract catchment average climate data usng `extractCatchmentData`

Usage

```
catchments
```

Format

An object of class `SpatialPolygonsDataFrame` with 2 rows and 1 columns.

See Also

`extractCatchmentData` for extracting catchment average climate data.

Examples

```
# Load example cacthment boundaries.
data("catchments")
```

extractCatchmentData *extractCatchmentData extracts catchment average climate data from netCDF files containing Australian climate data.*

Description

extractCatchmentData extracts the AWAP climate data for each point or polygon. For the latter, either the daily spatial mean and variance (or user defined function) of each climate metric is calculated or the spatial data is returned.

Usage

```
extractCatchmentData(
  ncdfFilename = file.path(getwd(), "AWAP.nc"),
  ncdfSolarFilename = file.path(getwd(), "AWAP_solar.nc"),
  extractFrom = as.Date("1900-01-01", "%Y-%m-%d"),
  extractTo = as.Date(Sys.Date(), "%Y-%m-%d"),
  getPrecip = TRUE,
```

```
  getTmin = TRUE,
  getTmax = TRUE,
  getVprp = TRUE,
  getSolarrad = TRUE,
  getET = TRUE,
  DEM = "",
  catchments = "",
  temporal.timestep = "daily",
  temporal.function.name = "mean",
  spatial.function.name = "var",
  interpMethod = "",
  ET.function = "ET.MortonCRAE",
  ET.Mortons.est = "potential ET",
  ET.Turc.humid = F,
  ET.timestep = "monthly",
  ET.interp_missing_days = T,
  ET.interp_missing_entries = T,
  ET.interp_abnormal = T,
  ET.constants = list()
)
```

Arguments

ncdfFilename	is a full file name (as string) to the netCDF file.
ncdfSolarFilename	is the full file name (as string) to the netCDF file.
extractFrom	is a date string specifying the start date for data extraction. The default is "1900-1-1".
extractTo	is a date string specifying the end date for the data extraction. The default is today's date as YYYY-MM-DD.
getPrecip	logical variable for extracting precipitation. Default is TRUE.
getTmin	logical variable for extracting Tmin. Default is TRUE.
getTmax	logical variable for extracting Tmax. Default is TRUE.
getVprp	logical variable for extracting vapour pressure. Default is TRUE.
getSolarrad	logical variable for extracting solar radiation. Default is TRUE.
getET	logical variable for calculating Morton's potential ET. Note, to calculate set getTmin=T, getTmax=T, getVprp=T and getSolarrad=T. Default is TRUE.
DEM	is either the full file name to a ESRI ASCII grid (as lat/long and using GDA94) or a raster class grid object. The DEM is used for the calculation of Morton's PET. The Australian 9 second DEM can be loaded using getDEM().
catchments	is either the full file name to an ESRI shape file of points or polygons (latter assumed to be catchment boundaries) or a shape file already imported using readShapeSpatial(). Either way the shape file must be in long/lat (i.e. not projected), use the ellipsoid GRS 80, and the first column should be a unique ID.

`temporal.timestep`
: character string for the time step of the output data. The options are `daily`, `weekly`, `monthly`, `quarterly`, `annual` or a user-defined index for, say, water-years (see `xts::apply.period`). The default is `daily`.

`temporal.function.name`
: character string for the function name applied to aggregate the daily data to `temporal.timestep`. Note, NA values are not removed from the aggregation calculation. If this is required then consider writing your own function. The default is `mean`.

`spatial.function.name`
: character string for the function name applied to estimate the daily spatial spread in each variable. If `NA` or `""` and `catchments` is a polygon, then the spatial data is returned. The default is `var`.

`interpMethod`
: character string defining the method for interpolating the gridded data (see `raster::extract`). The options are: `'simple'`, `'bilinear'` and `''`. The default is `''`. This will set the interpolation to `'simple'` when `catchments` is a polygon(s) and to `'bilinear'` when `catchments` are points.

`ET.function`
: character string for the evapotranspiration function to be used. The methods that can be derived from the AWAP data are are `ET.Abtew`, `ET.HargreavesSamani`, `ET.JensenHaise`, `ET.Makkink`, `ET.McGuinnessBordne`, `ET.MortonCRAE` , `ET.MortonCRWE`, `ET.Turc`. Default is `ET.MortonCRAE`.

`ET.Mortons.est`
: character string for the type of Mortons Et estimate. For `ET.MortonCRAE`, the options are `potential ET`,`wet areal ET` or `actual areal ET`. For `ET.MortonCRWE`, the options are `potential ET` or `shallow lake ET`. The default is `potential ET`.

`ET.Turc.humid`
: logical variable for the Turc function using the humid adjustment.See `ET.Turc`. For now this is fixed at `F`.

`ET.timestep`
: character string for the evapotranpiration time step. Options are `daily`, `monthly`, `annual` but the options are dependent upon the chosen `ET.function`. The default is `monthly`.

`ET.interp_missing_days`
: T or F, indicating if missing days should be interpolated for PET calculation. Default is `T`. See `ReadInputs`

`ET.interp_missing_entries`
: T or F, indicating if missing data entries should be interpolated for PET calculation. Default is `T`. See `ReadInputs`

`ET.interp_abnormal`
: T or F, indicating if abnormal valuses should be interpolated for PET calculation. Default is `T`. See `ReadInputs`

`ET.constants`
: list of constants from Evapotranspiration package required for ET calculations. To get the data use the command `data(constants)`. Default is `list()`.

Details

Daily data is extracted and can be aggregated to a weekly, monthly, quarterly, annual or a user-defined timestep using a user-defined funcion (e.g. sum, mean, min, max as defined by `temporal.function.name`).

The temporally aggreated data at each grid cell is then used to derive the spatial mean or the spatial variance (or any other function as defined by `spatial.function.name`).

The calculation of the spatial mean uses the fraction of each AWAP grid cell within the catchment polygon. The variance calculation (or user defined function) does not use the fraction of the grid cell and returns NA if there are <2 grid cells in the catchment boundary. Prior to the spatial aggregation, evapotranspiration (ET) can also calculated; after which, say, the mean and variance PET can be calculated.

The data extraction will by default be undertaken from 1/1/1900 to yesterday, even if the netCDF grids were only built for a subset of this time period. If the latter situation applies, it is recommended that the extraction start and end dates are input by the user.

The ET can be calculated using one of eight methods at a user defined calculation time-step; that is the `ET.timestep` defines the time step at which the estimates are serived and differs from the output timestep as defined by `temporal.function.name`). When `ET.timestep` is monthly or annual then the ET estimate is linearly interpolated to a daily time step (using zoo:na.spline()) and then constrained to >=0. In calculating ET, the input data is pre-processed using Evapotranspiration::ReadInputs() such that missing days, missing enteries and abnormal values are interpolated (by default) with the former two interpolated using the "DoY average", i.e. replacement with same day-of-the-year average. Additionally, when AWAP solar radiation is required for the ET function, data is only available from 1/1/1990. To derive ET values <1990, the average solar radiation for each day of the year from 1/1/990 to "extractTo" is derived (i.e. 365 values) and then applied to each day prior to 1990. Importantly, in this situation the estimates of ET <1990 are dependent upon the end date extracted. Re-running the estimation of ET with a later extractTo data will change the estimates of ET prior to 1990.

Also, when "catchments" is points (not polygons), then the netCDF grids are interpolate using bilinear interpolation of the closest 4 grid cells.

Lastly, data is extracted for all time points and no temporal infilling is undertaken if the grid cells are blank.

Value

When `catchments` are polygons and `spatial.function.name` is not `NA` or `""`, then the returned variable is a list variable containing two data.frames. The first is the areal aggreggated climate metrics named `catchmentTemporal.` with a suffix as defined by `temporal.function.name`). The second is the measure of spatial variability named `catchmentSpatial.` with a suffix as defined by `spatial.function.name`).

When `catchments` are polygons and `spatial.function.name` does equal `NA` or `""`, then the returned variable is a `sp::SpatialPixelsDataFrame` where the first colum is the catchment IDs and the latter columns are the results for each variable at each time point as defined by `temporal.timestep`.

When `catchments` are points, the returned variable is a data.frame containing daily climate data at each point.

See Also

`makeNetCDF_file` for building the NetCDF files of daily climate data.

Examples

```
# The example shows how to extract and save data.
# For an additional example see \url{https://github.com/peterson-tim-j/AWAPer/blob/master/README.md}
#---------------------------------------

# Set dates for building netCDFs and extracting data.
startDate = as.Date("2000-01-01","%Y-%m-%d")
endDate = as.Date("2000-02-28","%Y-%m-%d")

# Set names for netCDF files.
ncdfFilename = 'AWAPer_demo.nc'
ncdfSolarFilename = 'AWAPer_solar_demo.nc'

# Remove any existing netCDF demo files.
if (file.exists(ncdfFilename))
   is.removed = file.remove(ncdfFilename)
if (file.exists(ncdfSolarFilename))
   is.removed = file.remove(ncdfSolarFilename)

# Build netCDF grids and over a defined time period.

file.names = makeNetCDF_file(ncdfFilename=ncdfFilename,
             ncdfSolarFilename=ncdfSolarFilename,
             updateFrom=startDate, updateTo=endDate)

# Load example cacthment boundaries.
data("catchments")

# Get the constanrs required for ET estimation.
data(constants,package='Evapotranspiration')

# Download and import the Australian 9 second DEM.
# Note, the DEM only needs be downloaded if ET is be estimated.
DEM_9s = getDEM()

# Extract daily climate data (precip, Tmin, Tmax, VPD, ET).
# Note, the input "catchments" can also be a file to a ESRI shape file.
climateData = extractCatchmentData(ncdfFilename=file.names$ncdfFilename,
              ncdfSolarFilename=file.names$ncdfSolarFilename,
              extractFrom=startDate, extractTo=endDate,
              catchments=catchments,DEM=DEM_9s, ET.constants=constants)

# Extract the daily catchment average data.
climateDataAvg = climateData$catchmentTemporal.mean

# Extract the daily catchment variance data.
climateDataVar = climateData$catchmentSpatial.var

# Extract the monthly total precipitation.
monthlyPrecipData = extractCatchmentData(ncdfFilename=file.names$ncdfFilename,
              ncdfSolarFilename=file.names$ncdfSolarFilename,
              extractFrom=startDate, extractTo=endDate,
```

```
                catchments=catchments,
                getTmin = F, getTmax = F, getVprp = F, getSolarrad = F, getET = F,
                temporal.timestep = 'monthly', temporal.function.name = 'sum')

# Extract the monthly precip. sum data.
monthlyPrecipData.sum = monthlyPrecipData$catchmentTemporal.sum
```

`getDEM` *Downloads and imports Geoscience Australia 9s DEM.*

Description

`getDEM` get Australian 9s DEM.

Usage

```
getDEM(
  workingFolder = getwd(),
  urlDEM = getURLs()$DEM,
  DEMfilename = "dem-9s.asc",
  keepFiles = F
)
```

Arguments

`workingFolder`	is the file path (as string) in which to download the zip file. The default is `getwd()`.
`urlDEM`	URL to the folder containing the Geoscience Australia 9s DEM. The default is taken from `getURLs()$DEM`.
`DEMfilename`	is the file name for the DEM (as string). The default is `'dem-9s.asc'`.
`keepFiles`	is a logical scalar to keep the downloaded zip file and extracted DEM ASCII file. The default is `FALSE`.

Details

getDEM downloads the Geoscience Australia 9 second DEM and then imports the grid.

The DEM is required for the calculation of evaportranspiration within `extractCatchmentData`. For details of the DEM see https://www.data.gov.au/dataset/geodata-9-second-dem-and-d8-digital-elevation-mo

Value

A RasterLayer DEM for Asutralia.

See Also

`extractCatchmentData` for extracting catchment daily average and variance data.

Examples

```
# Download the DEM.

# Set file name for DEM file to the system temp. directory.
DEM.file = tempfile(fileext='.asc')

# Download and import the Australian 9 second DEM.
DEM_9s = getDEM(workingFolder=dirname(DEM.file),
         DEMfilename=basename(DEM.file))
```

`getURLs` *Get default URLs for loading data.*

Description

`getURLs` get URLS to AWAP and Australian 9s DEM.

Usage

```
getURLs()
```

Details

This function returns a list of default URLs used to download the AWAP and DEM data.

Value

A list variable of URLs as characters.

Examples

```
URLs = getURLs()
```

`makeNetCDF_file` *Build netCDF files of the Bureau of Meteorology (Australia) national gridded climate data.*

Description

`makeNetCDF_file` builds two netCDF files containing Australian climate data.

Usage

```
makeNetCDF_file(
  ncdfFilename = file.path(getwd(), "AWAP.nc"),
  ncdfSolarFilename = file.path(getwd(), "AWAP_solar.nc"),
  updateFrom = as.Date("1900-01-01", "%Y-%m-%d"),
  updateTo = as.Date(Sys.Date() - 1, "%Y-%m-%d"),
  workingFolder = getwd(),
  keepFiles = FALSE,
  compressionLevel = 5,
  urlPrecip = getURLs()$precip,
  urlTmin = getURLs()$Tmin,
  urlTmax = getURLs()$Tmax,
  urlVprp = getURLs()$vprp,
  urlSolarrad = getURLs()$solarrad
)
```

Arguments

ncdfFilename	is a file path (as string) and name to the netCDF file. The default file name and path is file.path(getwd(),'AWAP.nc').
ncdfSolarFilename	is the file path (as string) and name to the netCDF file. The default namefile and path file.path(getwd(),'AWAP_solar.nc').
updateFrom	is a date string specifying the start date for the AWAP data. If ncdfFilename and ncdfSolarFilename are specified and exist, then the netCDF grids will be updated with new data from updateFrom. To update the files from the end of the last day in the file set updateFrom=NA. The default is "1900-1-1".
updateTo	is a date string specifying the end date for the AWAP data. If ncdfFilename and ncdfSolarFilename are specified and exist, then the netCDF grids will be updated with new data to updateFrom. The default is yesterday's date as YYYY-MM-DD.
workingFolder	is the file path (as string) in which to download the AWAP grid files. The default is getwd().
keepFiles	is a logical scalar to keep the downloaded AWAP grid files. The default is FALSE.
compressionLevel	is the netCDF compression level between 1 (low) and 9 (high), and NA for no compression. Note, data extracion runtime may slightly increase with the level of compression. The default is 5.
urlPrecip	URL to the folder containing the AWAP daily precipittaion grids. The default is from getURLs()$precip.
urlTmin	URL to the folder containing the AWAP daily minimum temperature grids. The default is from getURLs()$Tmin.
urlTmax	URL to the folder containing the AWAP daily maximum temperature grids. The default is from getURLs()$Tmax.
urlVprp	URL to the folder containing the AWAP daily vapour pressure grids. The default is from getURLs()$vprp.

`urlSolarrad` URL to the folder containing the AWAP daily solar radiation grids. The default is from `getURLs()$solarrad`.

Details

makeNetCDF_file creates two netCDF files of daily climate data.

One of the netCDF files contains precipitation, minimum daily temperature, maximum daily temperature and vappour pressure. It should span from 1/1/1900 to today and requires ~20GB of hard-drive space (using default compression). The second netCDF file contains the solar radiation and started from 1/1/1990 and be ~24GB and spatial gaps are infilled using a 3x3 moving average repeated 3 times. To minimise the runtime in extracting data, both files should be stored locally and not on a network drive. Also, building the files requires installation of 7zip.

The climate data is sourced from the Bureau of Meteorology Australian Water Availability Project (http://www.bom.gov.au/jsp/awap/. For details see Jones et al. (2009).

The output from this function is required for all data extraction functions within this package and must be ran prior.

The function can be used to build netCDF files from stratch or to update existng netCDF files previously derived from this function. To not build or update a variable, set its respective URL to `NA`.

Value

A list variable containing the full file name to the AWAP data and the AWAP solar data.

References

David A. Jones, William Wang and Robert Fawcett, (2009), High-quality spatial climate data-sets for Australia, Australian Meteorological and Oceanographic Journal, 58 , p233-248.

See Also

`extractCatchmentData` for extracting catchment daily average and variance data.

Examples

```
# The example shows how to build the netCDF data cubes.
# For an additional example see \url{https://github.com/peterson-tim-j/AWAPer/blob/master/README.md}
#--------------------------------------

# Set dates for building netCDFs and extracting data.
startDate = as.Date("2000-01-01","%Y-%m-%d")
endDate = as.Date("2000-02-28","%Y-%m-%d")

# Set names for netCDF files (in the system temp. directory).
ncdfFilename = tempfile(fileext='.nc')
ncdfSolarFilename = tempfile(fileext='.nc')

# Build netCDF grids for all data but only over a defined time period.

file.names = makeNetCDF_file(ncdfFilename=ncdfFilename,
```

```
                ncdfSolarFilename=ncdfSolarFilename,
                updateFrom=startDate, updateTo=endDate)
```

Index