Package 'APSIM'

July 24, 2017

Type Package

Title General Utility Functions for the 'Agricultural Production Systems Simulator'

Version 0.9.2

Date 2017-07-24

Author Justin Fainges

Maintainer Justin Fainges <Justin.Fainges@csiro.au>

LazyData true

Description Contains functions designed to facilitate the loading and transformation of 'Agricultural Production Systems Simulator' output files <https://www.apsim.info>. Input meteorological data (also known as ``weather'' or ``met'') files can also be generated from user supplied data.

License GPL-3

Imports data.table (>= 1.9.4), lubridate (>= 1.3.3), stringr (>= 0.6.2), plyr (>= 1.8.1), sirad, methods, utils, RSQLite

URL https://www.apsim.info

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-07-24 03:28:49 UTC

R topics documented:

`apsim` *APSIM: A general utility package for the Agricultural Production Systems Simulator*

Description

The APSIM package provides a number of general purpose utilities designed to simplify the importation of simulation output files and to assist in the building of weather (met) files. Future development will include the ability to manipulate .apsim files in order to create large factorial simulations.

Import output files

`loadApsim` A general purpose function to load multiple APSIM results into a single data frame or data table. This function can also be used to read a single output file.

Manipulate and create met files

`prepareMet` can be used to generate an APSIM formatted met file from CSV, Excel or netCDF formats. `loadMet` will import an APSIM formatted met file into a custom metFile object. `writeMetFile` will write a completed metFile object to a .met file ready for use in APSIM.

`checkCont` *Checks a single year for continuity. Called from prepareMet.*

Description

This is an internal function used by prepareMet to check for continuity in a single year of a met file.

Usage

```
checkCont(data)
```

Arguments

`data`	A dataframe containing met file data.

Details

From tav_amp.for in the APSIM source code: One earth orbit around the sun does not take an integral number of days - 365 + a small part of a day. Since the Gregorian calendar year is measured as 365 days, a correction for this err is made every fourth year by adding one day to the length of the year. This correction is a little too much, thus in the centesimal years the correction is not made. However this over corrects, so in every fourth centesimal year, the correction of adding one day is made.

To summarise - If the year is divisible by 4 it is a leap year, unless it is a centesimal year, in which case it must be divisible by 400. i.e. it is a leap year if either of the conditions hold: (1) the year is divisible by 4 but not by 100; (2) the year is divisible by 400.

`checkMet` *Check for met file errors.*

Description

Checks for errors as described in Wall, B.H. "TAMET: Computer program for processing meteorological data." CSIRO Australia. Division of Tropical Crops and Pastures.Tropical Agronomy Technical Memorandum (1977): No. 4, 13p.

Usage

```
checkMet(met, lmint = -8, umint = 32, lmaxt = 10, umaxt = 50)
```

Arguments

`met`	Met file object.
`lmint`	Lower bound on minimum temperature.
`umint`	Upper bound on minimum temperature.
`lmaxt`	Lower bound on maximum temperature.
`umaxt`	Upper bound on maximum temperature.

Details

Errors checked include:

- Temperature discontinuites.
- Temperatures that are too high orlow.
- Evaporation that is too high or low.
- Radiation that is too high or low.

Note that issues found may not stop APSIM from running but might indicate an issue with the weather data. Warnings may not be applicable for very hot or cold climates.

Expects input in metFile object. Use prepareMet or loadMet first.

Examples

```
data(met)
checkMet(met)
```

`GetApsimNGTable` *Read APSIM Next Generation output files.*

Description

Reads APSIM NG .db files.

Usage

```
GetApsimNGTable(dbLoc, table)
```

Arguments

`dbLoc`	The location of the .db file.
`table`	The name of the table to read.

Details

This function will read a single table from a given APSIM NG .db file.

Examples

```
## Not run: GetApsimNGTable("c:/outputs/Wheat.db", "Results")
```

`getToken` *Extract a token from a vector of strings.*

Description

Extract a token from a character vector in a data frame containing a number of delimited tokens. Specialised version of str_split that works on vectors.

Usage

```
getToken(x, cname, pos, sep, colName)
```

Arguments

`x`	A data frame
`cname`	The name of the column to extract a token from.
`pos`	The position of the desired token in the token string.
`sep`	The character to use as a seperator (e.g. ":", ";", etc.) Can be multi-character.
`colName`	A name for the new column.

Details

By default loadApsim will automatically create columns from constants inside the output files. However sometimes you might want to extract values from different data such as the file name. This is easy to do if the data is a fixed width, but it can be difficult when it's of variable length and delimited. str_split() is the most common way to deal with this sort of data, but it doesn't work on a vector.

getToken takes a data frame, a column of type 'character', a seperator and a token position to return each token in the given position. It then appends the token as a new column.

`insertTavAmp` *Inserts Tav and Amp into a met object.*

Description

Amp is obtained by averaging the mean daily temperature of each month over the entire data period resulting in twelve mean temperatures, and then subtracting the minimum of these values from the maximum. Tav is obtained by averaging the twelve mean monthly temperatures.

Usage

```
insertTavAmp(met)
```

Arguments

`met`	A met file object where the tav and amp will be inserted.

Details

The original documentation for the stand alone Tav_Amp program can be found at http://www.apsim.info/Portals/0/OtherProducts/tav_amp.pdf.

Value

A met file object to which tav and amp has been added.

Examples

```
data(met)
insertTavAmp(met)
```

kingsData *Weather data from Kingsthorpe, Queensland, Australia.*

Description

A dataset containing a year of weather data from Kingsthorpe.

Format

A data frame containing 365 rows and 10 columns.

Source

internal

loadApsim *Read APSIM .out files.*

Description

Reads APSIM .out files.

Usage

```
loadApsim(dir = NULL, loadAll = TRUE, filter = "\\.out",
  returnFrame = TRUE, n = 0, fill = FALSE, addConstants = TRUE,
  skipEmpty = FALSE)
```

Arguments

dir	(optional) The directory to search for .out files. This is not recursive. If omitted, the current working directory will be used.
loadAll	If TRUE will load all files in dir. If FALSE, will load a single file specified by dir. Default is TRUE.
filter	A regular expression that matches the files to be loaded. Default is \\.out which filters for standard apsim output files.
returnFrame	Return the data as a data frame or data table. Default is TRUE, FALSE returns a data table.
n	Read the first n files. Good for testing/debugging. Default is 0 (read all files found).
fill	Where the number or names of columns is not consistent across files, fill missing columns with NA. Default is FALSE which will throw an error if the columns don't match across all files.

addConstants	Add any constants (such as ApsimVersion, Title or factor levels) found as extra columns. Default is TRUE.
skipEmpty	Silently skip empty output files. If false, empty files will cause an error. Default is FALSE.

Details

By default, this function will read in all output files in a directory and combine them into a single data table. Setting loadAll=FALSE will read a single file. If the path is omitted it will try to find the file in the current working directory. Constants in outputs will be added as data columns and output files with differing numbers of columns can also be imported although by default this will result in an error. Care should be taken when using this option as a different number of columns in output files usually means the data came from a different set of reports or simulations which may not be relevant to your analysis.

Examples

```
## Not run: loadApsim("c:/outputs") # load everything in the outputs directory
## Not run: loadApsim("c:/outputs/simulation.out", loadAll=FALSE)
# load a single file (note extension is required).
## End(Not run)
## Not run: loadApsim("c:/outputs", returnFrame=FALSE, fill=TRUE)
  # load everything in the outputs directory, fill any missing columns and return a data table.
## End(Not run)
```

---

loadMet *Read an APSIM formatted met file.*

Description

This function reads in a previously formatted met file. Can be useful for calculating tav and amp for a previously completed file or for reading into R for further analysis.

Usage

```
loadMet(f)
```

Arguments

f	The full path to the file to read.

Details

For reading in other formats see the Importing External Data section in prepareMet. See metFile for more information on the met object.

Value

A metFile object containing the read met file.

Examples

```
## Not run: loadMet("Weather.met")
```

`met` *A complete metFile example using data from Dalby, Queensland, Australia.*

Description

Three years of data from from Dalby including location coordinates, tav and amp data and units for the weather data

Format

See `metFile` for more information on the metFile class.

Source

internal

`metFile-class` *An S4 class used to store all information about a met file.*

Description

An S4 class used to store all information about a met file.

Slots

`const` A character vector containing constants that wilkl be written to the file. Format is 'variable = value'.

`lat` A length one numeric vector.

`lon` A length one numeric vector.

`tav` A length one numeric vector.

`amp` A length one numeric vector.

`units` A character vector containing the names of the columns in `data`.

`data` A data frame containing per day weather data.

prepareMet *Convert raw data to the correct APSIM met format.*

Description

prepareMet accepts a data frame containing met data and prepares it for writing to an APSIM formatted source file.

Usage

```
prepareMet(data, lat = stop("Latitude required."),
  lon = stop("Longitude required."),
  units = stop("Vector for met units required. See function help if part of table."),
  newNames = NULL, date.format = "AU")
```

Arguments

data	A data frame containing the data to prepare.
lat	Latitude in decimal degrees.
lon	Longitude in decimal degrees.
units	A character vector containing units for each column.
newNames	(optional) A vector of new column names.
date.format	(optional) A string containg the date format to use.

Details

It will generate year/day columns from an existing date column (and add the required units), check to ensure that dates are continuous and checks for the existence of required column names (year, day, radn, mint, maxt and rain).

It will interpolate 365 day leap years (e.g. no extra day from GCMs) and returns a metFile object that can be used with other APSIM functions.

Value

A metFile S4 class containing the prepared met data.

Importing External Data

prepareMet accepts a standard R data frame as an argument. As such, you can use any importation package that returns data in or can be coerced to a data frame. Some examples: #'

- Microsoft Excel files - readxl
- NetCDF - RNetCDF
- MySQL database - RMySQL
- Generic databases (including Microsoft SQL Server) - RODBC

Specifiying units for data

Each data column requires a unit in order to be valid. Units need to be enclosed in parentheses. For unitless values, use "()". Units can be specified by passing a 'units' vector to `prepareMet` (see example) or they may already be included in the data as would be seen in an APSIM output file. In this case, use `loadMet` instead.

Examples

```
data(Kingsthorpe)
newNames <-c("Date", "maxt", "mint", "rain", "evaporation", "radn", "vp", "Wind", "RH", "SVP")
units <- c("()", "(oC)", "(oC)", "(mm)", "(mm)", "(MJ/m^2/day)", "()", "()", "()", "()")
prepareMet(kingsData, -27.48, 151.81, newNames = newNames, units = units)
```

---

writeMetFile *Write a metFile object to disk in APSIM met format.*

Description

Takes a completed metFile object (generated via `prepareMet` or `loadMet`) and writes it to disk.

Usage

```
writeMetFile(fileName, met)
```

Arguments

`fileName`	The file name to write to.
`met`	The metFile object to write.

Details

Note the file will not be written if the met object is missing required information such as latitude, tav or amp. Note that while longitude is required in `prepareMet`, it is not strictly required by APSIM and thus is optional here. However, it is best practise to include it so it will remain mandatory when building a met file via `prepareMet`.

Index