Package ‘apsimx’

August 20, 2021

Title  Inspect, Read, Edit and Run 'APSIM' "Next Generation" and 'APSIM' Classic

Version  2.0

Description  The functions in this package inspect, read, edit and run files for 'APSIM' "Next Generation" ('JSON') and 'APSIM' "Classic" ('XML'). The files with an 'apsim' extension correspond to 'APSIM' Classic (7.x) - Windows only - and the ones with an 'apsimx' extension correspond to 'APSIM' "Next Generation".
For more information about 'APSIM' see (<https://www.apsim.info/>), and for 'APSIM' next generation (<https://apsimnextgeneration.netlify.app/>).

Depends  R (>= 3.5.0)

License  GPL-3

Encoding  UTF-8

VignetteBuilder  knitr

BugReports  https://github.com/femiguez/apsimx/issues

Imports  DBI, jsonlite, knitr, RSQLite, tools, utils, xml2

Suggests  BayesianTools, datasets, daymetr, FedData, ggplot2, GSODR, listviewer, maps, mvtnorm, nasapower, nloptr, raster, reactR, markdown, sensitivity, soilDB, sp, spData, sf

LazyData  true

RoxygenNote  7.1.1

NeedsCompilation  no

Author  Fernando Miguez [aut, cre] (<https://orcid.org/0000-0002-4627-8329>)

Maintainer  Fernando Miguez <femiguez@iastate.edu>

Repository  CRAN

Date/Publication  2021-08-20 15:50:02 UTC
R topics documented:

apsim ................................................................. 3
apsim.options ...................................................... 4
apsimx .............................................................. 5
apsimx.options ..................................................... 6
apsimx_example .................................................... 7
apsimx_filetype ................................................... 8
apsimx_options .................................................... 9
apsimx_soil_profile .............................................. 10
apsim_example ..................................................... 13
apsim_options ..................................................... 14
apsim_version ...................................................... 15
as_apsim_met ...................................................... 15
auto_detect_apsimx_examples ................................. 16
auto_detect_apsim_examples ................................. 17
check_apsim_met .................................................. 18
compare_apsim ..................................................... 18
doy2date ............................................................. 20
edit_apsim .......................................................... 21
edit_apsimx ........................................................ 22
edit_apsimx_batch ................................................. 24
edit_apsimx_replacement ....................................... 26
edit_apsimx_replace_soil_profile ........................... 30
dit_apsim_deep .................................................... 32
eit_apsim_xml ........................................................ 33
extract_values_apsimx .......................................... 35
get_apsimx_json ................................................... 35
get_daymet2_apsim_met ......................................... 36
get_daymet_apsim_met .......................................... 37
get_gsod_apsim_met .............................................. 39
get_iemre_apsim_met ............................................ 40
get_iem_apsim_met .............................................. 41
get_isric_soil_profile ......................................... 42
get_power_apsim_met .......................................... 44
get_ssurgo_soil_profile ....................................... 45
get_ssurgo_tables .............................................. 46
grep_json_list .................................................... 47
impute_apsim_met ................................................. 48
insert_replacement_node ...................................... 48
inspect_apsim ..................................................... 50
inspect_apsimx ................................................... 52
inspect_apsimx_json ............................................ 54
inspect_apsimx_replacement ................................ 55
inspect_apsim_xml .............................................. 57
mcmc.apsim.env .................................................. 59
mcmc.apsimx.env ............................................... 59
**apsim**

*Run an APSIM (7.x) ‘Classic’ simulation*

**Description**

Run apsim from R. It’s for Windows only. It uses ‘shell’.

**Usage**

```r
apsim(
  file = "",
  src.dir = ".",
  silent = FALSE,
  value = c("report", "all", "none"),
  cleanup = FALSE,
  simplify = TRUE
)
```
Arguments

file file name to be run (the extension .apsim is optional)
src.dir directory containing the .apsim file to be run (defaults to the current directory)
silent whether to print messages from apsim simulation
value how much output to return: option ‘report’ returns only the ‘main’ report component; option ‘all’ returns all components of the simulation; option ‘none’ runs simulation but does not return a data frame.
cleanup logical. Whether to delete the .out and .sum files generated by APSIM. Default is FALSE.
simplify whether to return a single data frame when multiple simulations are present. If FALSE it will return a list.

Details

Run an APSIM (7.x) ‘Classic’ Simulation

A valid apsim file can be run from within R. The main goal is to make running APSIM-X simple, especially for large scale simulations or parameter optimization

Value

This function returns a data frame with APSIM output, but it depends on the argument ‘value’ above.

Examples

## See function 'apsim_example'

apsim.options  Environment which stores APSIM options

Description

Environment which can store the path to the executable and where examples are located. Creating an environment avoids the use of global variables or other similar practices which would have possible undesiriable consequences.

Usage

apsim.options
Format

An object of class environment of length 3.

Details

Environment which stores APSIM options

Value

This is an environment, so nothing to return.

Examples

```r
## Not run:
names(apsim.options)
apsim.options(exe.path = "some-new-path-to-executable")
apsim.options$exe.path

## End(Not run)
```
apsimx.options

value

how much output to return:
option ‘report’ returns only the ‘main’ report component;
option ‘all’ returns all components of the simulation;
option ‘none’ does not create a data.frame but it generates the databases option
‘user-defined’ should be the name of a specific table

cleanup

logical. Whether to delete the .db file generated by APSIM-X. Default is FALSE

simplify

whether to return a single data frame when multiple reports are present. If
FALSE it will return a list.

xargs

extra arguments to be passed to the APSIM-X run. Use function xargs_apsimx.

Details
Run an APSIM-X Simulation
A valid apsimx file can be run from within R. The main goal is to make running APSIM-X simple,
especially for large scale simulations or parameter optimization

Value

a data frame with the ‘Report’ from the APSIM-X simulation. The return value depends on the
argument ‘value’ above.

Examples

## See function 'apsimx_example' and vignette 'apsimx'

apsimx.options

Environment which stores APSIM-X options

Description

Environment which can store the path to the executable, warning settings and where examples are
located. Creating an environment avoids the use of global variables or other similar practices which
would have possible undesirable consequences.

Usage

apsimx.options

Format

An object of class environment of length 5.

Details

Environment which stores APSIM-X options
**Value**

This is an environment, not a function, so nothing is returned.

**Examples**

```r
names(apsimx.options)
apsimx_options(exe.path = "some-new-path-to-executable")
apsimx.options$exe.path
```

---

**Description**

simple function to run some of the built-in APSIM-X examples

**Usage**

```r
apsimx_example(example = "Wheat", silent = FALSE)
```

**Arguments**

- `example`: run an example from built-in APSIM-X. Options are all of the ones included with the APSIM-X distribution, except ‘Graph’.
- `silent`: whether to print standard output from the APSIM-X execution

**Details**

This function creates a temporary copy of the example file distributed with APSIM-X to avoid writing a .db file to the directory where the ‘Examples’ are located. It is not a good practice and there is no guarantee that the user has read/write permissions in that directory.

**Value**

It returns a data frame

**Note**

This function creates a new column ‘Date’ which is in the R ‘Date’ format which is convenient for graphics.
Examples

```r
## Not run:
wheat <- apsimx_example("Wheat")
maize <- apsimx_example("Maize")
barley <- apsimx_example("Barley")
## The 'Date' column is created by this function, based on apsim output.
require(ggplot2)
ggplot(data = wheat, aes(x = Date, y = Yield)) +
  geom_point()
## End(Not run)
```

apsimx_filetype

Test file format for .apsimx files

Description

Test whether an .apsimx file is XML or json

Usage

```r
apsimx_filetype(file = "", src.dir = ".")
```

Arguments

- **file**: file ending in .apsimx to be tested
- **src.dir**: directory containing the .apsimx file to be tested; defaults to the current working directory

Value

`"xml"`, `"json"` or `"unknown"`

Note

Minimal function which reads only the first line in a file and tries to guess whether it is an `xml` or `json` file type.

Examples

```r
extd.dir <- system.file("extdata", package = "apsimx")
apsimx_filetype("Wheat.apsimx", src.dir = extd.dir)
```
Description

Set the path to the APSIM-X executable, examples and warning suppression.

Usage

```r
apsimx_options(
  exe.path = NA,
  examples.path = NA,
  warn.versions = TRUE,
  warn.find.apsimx = TRUE
)
```

Arguments

- `exe.path` path to apsim executable. White spaces are not allowed.
- `examples.path` path to apsim examples
- `warn.versions` logical. warning if multiple versions of APSIM-X are detected.
- `warn.find.apsimx` logical. By default a warning will be thrown if APSIM-X is not found. If `exe.path` is `NA` an error will be thrown instead.

Details

Set apsimx options

Value

as a side effect it modifies the `apsimx.options` environment.

Note

It is possible that APSIM-X is installed in some alternative location other than the defaults ones. Guessing this can be difficult and then the auto_detect functions might fail. Also, if multiple versions of APSIM-X are installed apsimx will choose the newest one but it will issue a warning. Suppress the warning by setting `warn.versions = FALSE`.

Examples

```r
names(apsimx.options)
apsimx_options(exe.path = "some-new-path-to-executable")
apsimx.options$exe.path
```
Create APSIM-X Soil Profiles

Description
Generates a soil profile that can then replace the existing one in an `.apsimx` or `.apsim` simulation file.

plotting function for a soil profile, it requires `ggplot2`

checking an apsimx soil profile for reasonable values

Usage

```r
apsimx_soil_profile(
  nlayers = 10,
  Depth = NULL,
  Thickness = NULL,
  BD = NULL,
  AirDry = NULL,
  LL15 = NULL,
  DUL = NULL,
  SAT = NULL,
  KS = NULL,
  crop.LL = NULL,
  crop.KL = NULL,
  crop.XF = NULL,
  Carbon = NULL,
  SoilCNRatio = NULL,
  FOM = NULL,
  FOM.CN = NULL,
  FBiom = NULL,
  FInert = NULL,
  NO3N = NULL,
  NH4N = NULL,
  PH = NULL,
  soil.bottom = 150,
  water.table = 200,
  soil.type = 0,
  crops = c("Maize", "Soybean", "Wheat"),
  metadata = NULL,
  soilwat = NA,
  swim = NA,
  dist.parms = list(a = 0, b = 0.2)
)
```

```r
## S3 method for class 'soil_profile'
plot()
```
check_apsimx_soil_profile(x)

Arguments

- `nlayers`: Number of soil layers (default = 10)
- `Depth`: specific depths for each soil layer (cm)
- `Thickness`: thickness for each soil layer (mm)
- `BD`: bulk density for each soil layer (g/cc) – ‘cc’ is cubic cm
- `AirDry`: air dry for each soil layer (mm/mm)
- `LL15`: lower limit (15 bar) for each soil layer (mm/mm)
- `DUL`: drainage upper limit (0.33 bar) for each soil layer (mm/mm)
- `SAT`: saturation (0 bar) for each soil layer (mm/mm)
- `KS`: saturated hydraulic conductivity (mm/day)
- `crop.LL`: lower limit for a specific crop
- `crop.KL`: root ability to extract water for a specific crop
- `crop.XF`: soil root exploration for a specific crop
- `Carbon`: organic carbon (percent)
- `SoilCNRatio`: organic carbon C:N ratio
- `FOM`: fresh organic matter (kg/ha)
- `FOM.CN`: fresh organic matter C:N ratio
- `FBiom`: Fraction of microbial biomass (0-1)
- `FInert`: Fraction of inert carbon (0-1)
- `NO3N`: nitrate nitrogen (Chemical) (ppm)
- `NH4N`: ammonium nitrogen (Chemical) (ppm)
- `PH`: soil pH
- `soil.bottom`: bottom of the soil profile (cm)
- `water.table`: water table level (not used at the moment) (cm)
- `soil.type`: might use it in the future for auto filling missing information
- `crops`: name of crops being grown
- `metadata`: list with soil metadata. For possible parameters and values see an example of `inspect_apsimx` with soil.child = “Metadata”.
- `soilwat`: optional ‘list’ of class ‘soilwat_parms’
- `swim`: optional ‘list’ of class ‘swim_parms’
apsimx_soil_profile

dist.parms parameter values for creating a profile. If a == 0 and b == 0 then
a constant value of 1 is used. If a == 0 and b != 0, then an exponential decay is
used.
If a != 0 and b != 0 then the equation is a*soil.layer*exp(-b*soil.layer).

x object of class ‘soil_profile’ or the ‘soil’ component within an object of class
‘soil_profile’.

... additional plotting arguments (none use at the moment).

property “all” for plotting all soil properties, “water” for just SAT, DUL and LL15

Details

Soil Profiles

Real soils might have discontinuities, but for APSIM it might be beneficial to be able to create a
soil profile with an arbitrary number of layers and have flexibility in the distribution of soil physical
and chemical properties. Steps:

1. apsimx_soil_profile is a function which can create a soil matrix with many layers
2. It allows for creating a smooth distribution for Physical (or Water), Chemical, InitialWater, Analysis, InitialN, Organic or SoilOrganicMatter
3. The distribution can be specified with the ‘a’ and ‘c’ parameter of an exponential decay function,
   using a list. E.g. DUL = list(0.35, 0, -0.1). This means that the top value for DUL will be 0.35 and
   it will decay with a rate of -0.1.
4. If an increase and then a decay is needed the Ricker function can be used. See ‘SSricker’ in the
   ‘nlraa’ package.

Value

a soil profile with class ‘soil_profile’ with elements ‘soil’, ‘crops’, ‘metadata’, ‘soilwat’ and ‘swim’.
it produces a plot

It does not produce output unless potential issues are found. Only warnings are produced and it
returns an object of class ‘soil_profile’.

Examples

sp <- apsimx_soil_profile()
require(ggplot2)
plot(sp)
**Description**

simple function to run some of the built-in APSIM examples

**Usage**

```r
apsim_example(example = "Millet", silent = FALSE, tmp.dir = NULL)
```

**Arguments**

- `example`: run an example from built-in APSIM. Options are all of the ones included with the APSIM distribution, except ‘Graph’.
- `silent`: whether to print standard output from the APSIM execution
- `tmp.dir`: temporary directory where to write files

**Details**

This function creates a temporary copy of the example file distributed with APSIM to avoid writing a .out file to the directory where the ‘Examples’ are located. It is not a good practice and there is no guarantee that the user has read/write permissions in that directory.

**Value**

This function returns a data frame with APSIM output

**Note**

This function creates a new column ‘Date’ which is in the R ‘Date’ format which is convenient for graphics.

**Examples**

```r
## Not run:
## Only run these if you have APSIM 'Classic' installed (Windows only)
millet <- apsim_example("Millet")
potato <- apsim_example("Potato")
sugar <- apsim_example("Sugar")
## The 'Date' column is created by this function, based on apsim output.
require(ggplot2)
ggplot(data = millet, aes(x = Date, y = millet_biomass)) +
  geom_line()
```

## End(Not run)
apsim_options  Setting some options specific to APSIM (7.x) ‘Classic’

Description

Set the path to the APSIM executable, examples and warning suppression.

Usage

```r
apsim_options(exe.path = NA, examples.path = NA, warn.versions = TRUE)
```

Arguments

- `exe.path` path to apsim executable
- `examples.path` path to apsim examples
- `warn.versions` logical. warning if multiple versions of APSIM are detected.

Details

Set apsim options

Value

It modifies the ‘apsim.options’ environment as a side effect.

Note

It is possible that APSIM 7.x ‘Classic’ is installed in some alternative location other than the defaults ones. Guessing this can be difficult and then the auto_detect functions might fail. Also, if multiple versions of APSIM are installed apsim will choose the newest one but it will issue a warning. Suppress the warning by setting `warn.versions = FALSE`.

Examples

```r
## Not run:
names(apsim.options)
apsim_options(exe.path = "some-new-path-to-executable")
apsim.options$exe.path

## End(Not run)
```
**apsim_version**  
*Display available APSIM ‘Classic’ and APSIM-X versions*

**Description**  
Display available APSIM ‘Classic’ and APSIM-X versions

**Usage**  
`apsim_version(which = c("all", "inuse"), verbose = TRUE)`

**Arguments**  
- `which` either ‘all’ or ‘inuse’  
- `verbose` whether to print the information to standard output

**Value**  
a data frame (all) or a vector (inuse) with APSIM-X and/or APSIM versions

**Examples**  
```r  
## Not run:  
## Check which apsim version are available  
ava <- apsim_version(verbose = TRUE)  
## End(Not run)
```

**as_apsim_met**  
*Conversion from data frame to met object*

**Description**  
It makes minimum assumptions about the data so it is recommended to change defaults

**Usage**  
`as_apsim_met(`  
```
x,  
filename = "noname.met",  
site = "nosite",  
latitude = 0,  
longitude = 0,  
tav = NA,  
amp = NA,
```
auto_detect_apsimx_examples

Arguments

x object of class ‘data frame’
filename default ‘noname.met’
site default ‘nosite’
latitude default is zero (0)
longitude default is zero (0)
tav average temperature (calculated if not supplied)
amplitude temperature amplitude (calculated if not supplied)
colnames default are “year”, “day”, “radn”, “maxt”, “mint”, “rain”
units default are “()”, “()”, “(MJ/m2/day)”, “(oC)”, “(oC)”, “(mm)”
constants default is “NA”
comments default is “NA”
check whether to check the resulting met file using check_apsim_met. default is TRUE.

Details

Simple utility for converting a data frame to an object of class met

Value

it returns an object of class ‘met’.

Description

simple function to detect where APSIM-X examples are located

Usage

auto_detect_apsimx_examples()
auto_detect_apsim_examples

Details
Auto detect where apsimx examples are located

Value
will create a directory (character string) pointing to APSIM-X distributed examples

Examples

```r
## Not run:
ex.dir <- auto_detect_apsimx_examples()
## End(Not run)
```

Description
simple function to detect where APSIM ‘Classic’ examples are located

Usage
center

auto_detect_apsim_examples()

details
Auto detect where APSIM (7.x) ‘Classic’ examples are located

Value
will create a directory pointing to APSIM ‘Classic’ distributed examples

Examples

```r
## Not run:
ex.dir <- auto_detect_apsim_examples()
## End(Not run)
```
**check_apsim_met**  
*Check a met file for possible errors*

**Description**
Takes in an object of class ‘met’ and checks for missing/valid/reasonable values

**Usage**
```
check_apsim_met(met)
```

**Arguments**
- `met`: object of class ‘met’

**Details**
- It will only check for missing values and reasonable (within range) values for: ‘year’: range (1500 to 3000); ‘day’: range (1 to 366); ‘maxt’: range (-60 to 60) – units (C); ‘mint’: range (-60 to 40) – units (C); ‘radn’: range (0 to 40) – units (MJ/m2/day); ‘rain’: range (0 to 100) – units (mm/day)

**Value**
does not return anything unless possible errors are found

---

**compare_apsim**  
*Compare two or more apsim output objects*

**Description**
Function which allows for a simple comparison between APSIM output objects. plotting function for compare_apsim, it requires ggplot2

**Usage**
```
compare_apsim(..., variable, index = "Date", by, labels)
```

```
## S3 method for class 'out_mrg'
plot(
  x,
  ..., 
)```
plot.type = c("vs", "diff", "ts", "density"),
pairs = c(1, 2),
cumulative = FALSE,
variable,
id,
span = 0.75
)

Arguments

... data frames with APSIM output or observed data.
variable variable to plot
index index for merging objects. Default is ‘Date’
by factor for splitting the comparison, such as a treatment effect.
labels labels for plotting and identification of objects.
x object of class ‘out_mrg’
plot.type either ‘vs’, ‘diff’, ‘ts’ - for time series or ‘density’
pairs pair of objects to compare, defaults to 1 and 2 but others are possible
cumulative whether to plot cumulative values (default FALSE)
id identification (not implemented yet)
span argument passed to ‘geom_smooth’

Value

object of class ‘out_mrg’, which can be used for further plotting
it produces a plot

Note

‘Con Corr’ is the concordance correlation coefficient (https://en.wikipedia.org/wiki/Concordance_correlation_coefficient);
‘ME’ is the model efficiency (https://en.wikipedia.org/wiki/Nash

Examples

## Directory with files
extd.dir <- system.file("extdata", package = "apsimx")
## Comparing observed and simulated for Wheat
data(obsWheat)
sim.opt <- read.csv(file.path(extd.dir, "wheat-sim-opt.csv"))
sim.opt$Date <- as.Date(sim.opt$Date)
cap <- compare_apsim(obsWheat, sim.opt, labels = c("obs", "sim"))
plot(cap)
plot(cap, plot.type = "diff")
plot(cap, plot.type = "ts")


plot(cap, variable = "AboveGround")
plot(cap, variable = "AboveGround", plot.type = "diff")
plot(cap, variable = "AboveGround", plot.type = "ts")

---

**compare_apsim_met**  
*Compare two or more metfiles*

**Description**

Helper function which allows for a simple comparison among help files plotting function for compare_apsim_met, it requires ggplot2

**Usage**

```r
compare_apsim_met(
    ...,
    met.var = c("all", "radn", "maxt", "mint", "rain", "rh", "wind_speed", "vp"),
    labels,
    check = FALSE
)
```

```r
## S3 method for class 'met_mrg'
plot(
    x,
    ...
    plot.type = c("vs", "diff", "ts", "density"),
    pairs = c(1, 2),
    cumulative = FALSE,
    met.var = c("radn", "maxt", "mint", "rain"),
    id,
    span = 0.75
)
```

**Arguments**

- **...** met file objects. Should be of class ‘met’
- **met.var** meteorological variable to plot
- **labels** labels for plotting and identification of ‘met’ objects.
- **check** whether to check met files using ‘check_apsim_met’.
- **x** object of class ‘met_mrg’
- **plot.type** either ‘vs’, ‘diff’, ‘ts’ - for time series or ‘density’
- **pairs** pair of objects to compare, defaults to 1 and 2 but others are possible
**doy2date**

Converts from doy to date

**Description**

Given a day of the year as julian (1-366) it converts to ‘Date’

Given a ‘Date’ it converts to julian day (1-366) or day of the year

cumulative  whether to plot cumulative values (default FALSE)
id        identification (not implemented yet)
span      argument to be passed to ‘geom_smooth’

**Value**

object of class ‘cmet’, which can be used for further plotting

it produces a plot

**Note**

I have only tested this for 2 or 3 objects. The code is set up to be able to compare more, but I’m not sure that would be all that useful.

**Examples**

```r
## Not run:
require(nasapower)
## Specify the location
lonlat <- c(-93, 42)
## dates
dts <- c("2017-01-01","2017-12-31")
## Get pwr
pwr <- get_power_apsim_met(lonlat = lonlat, dates = dts)
## Get data from IEM
iem <- get_iem_apsim_met(lonlat = lonlat, dates = dts)
## Compare them
cmet <- compare_apsim_met(pwr[,1:6], iem, labels = c("pwr","iem"))
## Visualize radiation
plot(cmet, met.var = "radn")
plot(cmet, plot.type = "diff")
plot(cmet, plot.type = "ts")
## Visualize maxt
plot(cmet, met.var = "maxt")
plot(cmet, met.var = "maxt", plot.type = "diff")
plot(cmet, met.var = "maxt", plot.type = "ts")
## Cumulative rain
plot(cmet, met.var = "rain", plot.type = "ts", cumulative = TRUE)
```

```r
## End(Not run)
```
Usage

doy2date(x, year = 2001, inverse = FALSE)

date2doy(x, year = 2001, inverse = FALSE)

Arguments

x either an integer 1-366 or a ‘Date’
year year
inverse if TRUE it goes from doy to ‘Date’

Value

an object of class ‘Date’ or a numeric if inverse equals TRUE.
an numeric or an object of class ‘Date’ if inverse equals TRUE.

Examples

doy2date(120)
date2doy("04-30")

edit_apsim Edit an APSIM (Classic) Simulation

Description

This function allows editing of an APSIM (Classic) simulation file.

Usage

edit_apsim(
  file,
  src.dir = ".", wrt.dir = NULL,
            "Manager", "Outputfile", "Other"),
                "Analysis", "InitialWater", "Sample", "SWIM"),
  manager.child = NULL,
  parm = NULL,
  value = NULL,
  overwrite = FALSE,
  edit.tag = "-edited",
  parm.path = NULL,
  root,
Syntax:
```
edit_apsim = TRUE,
check.length = TRUE
```

**Arguments**

- **file**: file ending in .apsim to be edited
- **src.dir**: directory containing the .apsim file to be edited; defaults to the current working directory
- **wrt.dir**: should be used if the destination directory is different from the src.dir
- **soil.child**: specific soil component to be edited
- **manager.child**: specific manager component to be edited (not implemented yet)
- **parm**: parameter to be edited
- **value**: new values for the parameter to be edited
- **overwrite**: logical; if TRUE the old file is overwritten, a new file is written otherwise
- **edit.tag**: if the file is edited a different tag from the default ‘-edited’ can be used.
- **parm.path**: path to the attribute to edit when node is ‘Other’
- **root**: supply the node position in the case of multiple simulations such as factorials.
- **verbose**: whether to print information about successful edit
- **check.length**: check whether vectors are of the correct length

**Details**

The variables specified by **parm** within the .apsim file specified by **file** in the source directory **src.dir** are edited. The old values are replaced with **value**, which is a list that has the same number of elements as the length of the vector **parm**. The current .apsim file will be overwritten if **overwrite** is set to TRUE; otherwise the file ‘file’ ‘-edited.apsim will be created. If (verbose = TRUE) then the name of the written file is returned.

When node equals Outputfile, the editing allows to add variables, but not to remove them at the moment.

**Value**

(when verbose=TRUE) complete file path to edited .apsimx file is returned as a character string. As a side effect this function creates a new (XML) .apsimx file.

**Note**

The components that can be edited are restricted because this is better in preventing errors of editing unintended parts of the file. The disadvantage is that there is less flexibility compared to the similar function in the ‘apsimr’ package.
Examples

```r
## This example will read one of the examples distributed with APSIM
## but write to a temporary directory

tmp.dir <- tempdir()

extd.dir <- system.file("extdata", package = "apsimx")
edit_apsim("Millet", src.dir = extd.dir, wrt.dir = tmp.dir,
  node = "Clock",
  parm = "start_date", value = "01/02/1940")

## Editing all of the KL values for Millet
pp.KL <- inspect_apsim_xml("Millet.apsim", src.dir = extd.dir,
  parm = "SoilCrop[8]/KL")

kls <- seq(0.08, 0.2, length.out = 11)

edit_apsim("Millet.apsim",
  src.dir = extd.dir,
  wrt.dir = tmp.dir,
  node = "Other",
  parm.path = pp.KL,
  value = kls)

## Check that it was properly edited
inspect_apsim("Millet-edited.apsim",
  src.dir = tmp.dir,
  node = "Soil",
  soil.child = "Water",
  parm = "KL")
```

edit_apsimx  
**Edit an APSIM-X (JSON) Simulation**

Description

This function allows editing of an APSIM-X (JSON) simulation file.

Usage

```r
edit_apsimx(
  file,
  src.dir = ".",
```
Arguments

- **file**
  - File ending in .apsimx to be edited (JSON).
- **src.dir**
  - Directory containing the .apsimx file to be edited; defaults to the current working directory.
- **wrt.dir**
  - Should be used if the destination directory is different from the src.dir.
- **node**
  - Either 'Clock', 'Weather', 'Soil', 'SurfaceOrganicMatter', 'MicroClimate', 'Crop', 'Manager', 'Report' or 'Other'.
- **soil.child**
  - Specific soil component to be edited.
- **manager.child**
  - Specific manager component to be edited.
- **parm**
  - Parameter to be edited.
- **value**
  - New values for the parameter to be edited.
- **overwrite**
  - Logical; if TRUE the old file is overwritten, a new file is written otherwise.
- **edit.tag**
  - If the file is edited a different tag from the default '-edited' can be used.
- **parm.path**
  - Path to the attribute to edit when node is 'Other'.
- **root**
  - Supply the node position in the case of multiple simulations such as factorials.
- **verbose**
  - Whether to print information about successful edit.

Details

The variables specified by **parm** within the .apsimx file specified by **file** in the source directory **src.dir** are edited. The old values are replaced with **value**, which is a list that has the same number of elements as the length of the vector **parm**. The current .apsimx file will be overwritten if **overwrite** is set to TRUE; otherwise the file 'file'-*edited.apsimx* will be created. If (verbose = TRUE) then the name of the written file is returned.

When node equals Report, the editing allows to add variables, but not to remove them at the moment.

Value

(when verbose=TRUE) Complete file path to edited .apsimx file is returned as a character string. As a side effect this function creates a new (JSON) .apsimx file.
Examples

```r
## This example will read one of the examples distributed with APSIM-X
## but write to a temporary directory
tmp.dir <- tempdir()

## Edit Bulk density
text.dir <- system.file("extdata", package = "apsimx")
bds <- c(1.02, 1.03, 1.09, 1.16, 1.18, 1.19, 1.20)
edition_apsimx("Wheat.apsimx", src.dir = extd.dir,
                   wrt.dir = tmp.dir,
                   node = "Soil",
                   soil.child = "Water",
                   parm = "BD", value = bds,
                   verbose = FALSE)

## Inspect file
inspect_apsimx("Wheat-edited.apsimx", src.dir = tmp.dir,
               node = "Soil", soil.child = "Water")

## To delete the file...
file.remove(file.path(tmp.dir, "Wheat-edited.apsimx"))

## Edit the fertilizer amount in 'Maize.apsimx'
edition_apsimx("Maize.apsimx", src.dir = extd.dir,
                wrt.dir = tmp.dir, node = "Manager",
                manager.child = "SowingFertiliser",
                parm = "Amount", value = 200, verbose = TRUE)

## Make sure it worked
inspect_apsimx("Maize-edited.apsimx", src.dir = tmp.dir, node = "Manager")

## Remove the file
file.remove(file.path(tmp.dir, "Maize-edited.apsimx"))
```

edit_apsimx_batch

Edit an APSIM-X (JSON) Simulation in Batch mode

Description

This function allows editing of an APSIM-X (JSON) simulation file in batch mode.

Usage

```r
edit_apsimx_batch(file = NULL,
                    src.dir = ".",
                    wrt.dir = NULL,
                    parms = NULL,
                    ...)```
Arguments

- **file**: file ending in .apsimx to be edited (JSON)
- **src.dir**: directory containing the .apsimx file to be edited; defaults to the current working directory
- **wrt.dir**: should be used if the destination directory is different from the src.dir
- **parms**: parameter to be edited in the form 'key = value'
- **silent**: controls the output of running APSIM at the command line
- **verbose**: whether to print information about successful edit

Details

from hol430

This allows the user to specify an .apsimx file and a config file when running Models.exe. The .apsimx file will not be run but instead, the changes listed in the config file will be applied to the .apsimx file, which will then be written to disk under the same filename.

The config file should contain lines of the form 'path = value'

E.g.:

```
```

Command line arguments should look like: Models.exe file.apsimx /Edit /path/to/config/file.conf

Relative paths will be resolved to the first match. i.e., [Clock].StartDate will match the first clock found in the file.

Dates can be specified as yyyy-mm-dd or mm/dd/yyyy.

Strings should not be quoted

Array indices will be interpreted as 1-indexed (mad face). So the first element in the array should have index 1 in the config file.

The file will be upgraded to the latest file version as part of this process.

Value

(when verbose=TRUE) complete file path to edited .apsimx file is returned as a character string. As a side effect this function creates a new (JSON) .apsimx file.

Examples

```r
## This example will read one of the examples distributed with APSIM-X
## but write to a temporary directory

tmp.dir <- tempdir()
```
## Edit InitialResidueMass

```r
extd.dir <- system.file("extdata", package = "apsimx")
edit_apsimx_batch("Wheat.apsimx", src.dir = extd.dir, wrt.dir = tmp.dir, parms = parms)
```

---

**edit_apsimx_replacement**

_Edit a replacement component in an .apsim (JSON) file_

### Description

edit the replacement component of an JSON apsimx file. It does not replace the GUI, but it can save time by quickly editing parameters and values.

### Usage

```r
edit_apsimx_replacement(
  file = "", # file ending in .apsimx to edit (JSON)
  src.dir = ".", # directory containing the .apsimx file; defaults to the current working directory
  wrt.dir = ".", # should be used if the destination directory is different from the src.dir
  node = NULL, # specific node to edit
  node.child = NULL, # specific node child component to edit.
  root = list("Models.Core.Replacements", NA),
  parm = NULL,
  value = NULL,
  overwrite = FALSE,
  edit.tag = "-edited",
  verbose = TRUE
)
```

### Arguments

- `file` file ending in .apsimx to edit (JSON)
- `src.dir` directory containing the .apsimx file; defaults to the current working directory
- `wrt.dir` should be used if the destination directory is different from the src.dir
- `node` specific node to edit
- `node.child` specific node child component to edit.
node.subchild  specific node sub-child to edit.
node.subsubchild specific node sub-subchild to edit.
node.sub3child  specific node sub-sub-subchild to edit.
node.sub4child  specific node sub-sub-sub-subchild to edit.
node.sub5child  specific node sub-sub-sub-sub-subchild to edit.
node.string    passing of a string instead of the node hierarchy. It can either start with a dot or not. However, the ‘best’ form is not to start with a dot as it should be a more convenient form of passing the nodes and their childs and not a real ‘jsonpath’.
root         ‘root’ node to explore (default = “Models.Core.Replacements”)
parm          specific parameter to edit
value         new values for the parameter
overwrite     logical; if TRUE the old file is overwritten, a new file is written otherwise
edit.tag      if the file is edited a different tag from the default ‘-edited’ can be used.
verbose       whether to print information about successful edit

Details

This is simply a script that prints the relevant parameters which are likely to need editing. It does not print all information from an .apsimx file.

Value

(when verbose=TRUE) complete file path to edited .apsimx file is returned as a character string. As a side effect this function creates a new (JSON) .apsimx file.

Note

The components that can be edited are restricted becuase this is better in preventing errors of editing unintended parts of the file.

Examples

```
extd.dir <- system.file("extdata", package = "apsimx")
## Writing to a temp directory, but change as needed
tmp.dir <- tempdir()

## Inspect original values
inspect_apsimx_replacement("MaizeSoybean.apsimx",
    src.dir = extd.dir,
    node = "Maize",
    node.child = "Phenology",
    node.subchild = "ThermalTime",
    node.subsubchild = "BaseThermalTime",
    node.sub3child = "Response")```
```
edit_apsimx_replacement("MaizeSoybean.apsimx",
    src.dir = extd.dir, wrt.dir = tmp.dir,
    node = "Maize",
    node.child = "Phenology",
    node.subchild = "ThermalTime",
    node.subsubchild = "BaseThermalTime",
    node.sub3child = "Response",
    parm = "X",
    value = c(10, 20, 30, 40, 50))
## inspect it
inspect_apsimx_replacement("MaizeSoybean-edited.apsimx",
    src.dir = tmp.dir,
    node = "Maize",
    node.child = "Phenology",
    node.subchild = "ThermalTime",
    node.subsubchild = "BaseThermalTime",
    node.sub3child = "Response")
## Illustrating using 'node.string'
## Equivalent to the code to edit above
edit_apsimx_replacement("MaizeSoybean-edited.apsimx",
    src.dir = tmp.dir, wrt.dir = tmp.dir,
    parm = "X",
    value = c(11, 21, 31, 41, 51),
    edit.tag = "-ns")
inspect_apsimx_replacement("MaizeSoybean-edited-ns.apsimx",
    src.dir = tmp.dir,
    node = "Maize",
    node.child = "Phenology",
    node.subchild = "ThermalTime",
    node.subsubchild = "BaseThermalTime",
    node.sub3child = "Response")
```

---

**edit_apsimx_replace_soil_profile**

*Edit APSIM-X file with a replaced soil profile*

**Description**

Edits an APSIM-X simulation by replacing the soil profile

**Usage**

```
edit_apsimx_replace_soil_profile(
    file = "",
```
**edit_apsimx_replace_soil_profile**

```r
src.dir = ".",
wrt.dir = NULL,
soil.profile = NULL,
edit.tag = "-edited",
overwrite = FALSE,
verbose = TRUE
)
```

**Arguments**

- **file**: name of the .apsimx file to be edited
- **src.dir**: source directory
- **wrt.dir**: writing directory
- **soil.profile**: a soil profile object with class ‘soil_profile’
- **edit.tag**: default edit tag ‘-edited’
- **overwrite**: default FALSE
- **verbose**: default TRUE and it will print messages to console

**Details**

This function is designed to batch replace the whole soil in an APSIM simulation file.

**Value**

writes a file to disk with the supplied soil profile

**Note**

There is no such thing as a default soil, carefully build the profile for each simulation.

**Examples**

```r
sp <- apsimx_soil_profile()
extd.dir <- system.file("extdata", package = "apsimx")

## I write to a temp directory but replace as needed
tmp.dir <- tempdir()

edit_apsimx_replace_soil_profile("Maize.apsimx", soil.profile = sp,
                                 src.dir = extd.dir, wrt.dir = tmp.dir)
inspect_apsimx("Maize-edited.apsimx", src.dir = tmp.dir,
               node = "Soil")
```
**Description**

Edits an APSIM Classic simulation by replacing the soil profile.

**Usage**

```r
edit_apsim_replace_soil_profile(
  file = "",
  src.dir = ".",
  wrt.dir = NULL,
  soil.profile = NULL,
  swim = NULL,
  soilwat = NULL,
  edit.tag = "-edited",
  overwrite = FALSE,
  verbose = TRUE
)
```

**Arguments**

- `file`: name of the .apsim file to be edited.
- `src.dir`: source directory.
- `wrt.dir`: writing directory.
- `soil.profile`: a soil profile object with class 'soil_profile'.
- `swim`: list with SWIM specific parameters.
- `soilwat`: list with SoilWat specific parameters.
- `edit.tag`: default edit tag '-edited'.
- `overwrite`: default FALSE.
- `verbose`: default TRUE. Will print messages indicating what was done.

**Details**

This function is designed to batch replace the whole soil in an APSIM simulation.

**Value**

writes an APSIM file to disk with the supplied soil profile.
Note

There is no such thing as a default soil, carefully build the profile for each simulation. This function replaces values and it can grow an XML node, but it cannot edit a property which is not present in the original file.

Examples

```r
sp <- apsimx_soil_profile(nlayers = 20,
  crops = c("Barley", "Chickpea", "Lucerne",
   "Maize", "Perennial Grass", "Sorghum",
   "Wheat", "Millet"))

extd.dir <- system.file("extdata", package = "apsimx")

## Writing to a temp directory
tmp.dir <- tempdir()
edit_apsim_replace_soil_profile("Millet.apsim", soil.profile = sp,
  edit.tag = "-newsoil",
  src.dir = extd.dir,
  wrt.dir = tmp.dir)

inspect_apsim("Millet-newsoil.apsim", src.dir = tmp.dir,
  node = "Soil", soil.child = "Water")
```

edit_apsim_xml  Edit an APSIM (Classic) Simulation auxiliary xml file

Description

This function allows editing of an APSIM (Classic) simulation xml file.

Usage

```r
edit_apsim_xml(
  file,
  src.dir = ".",
  wrt.dir = NULL,
  parm.path = NULL,
  value = NULL,
  overwrite = FALSE,
  edit.tag = "-edited",
  verbose = TRUE
)
```
**Arguments**

- **file**: file ending in .xml to be edited
- **src.dir**: directory containing the .xml file to be edited; defaults to the current working directory
- **wrt.dir**: should be used if the destination directory is different from the src.dir
- **parm.path**: parameter path to be edited (see example)
- **value**: new values for the parameter to be edited
- **overwrite**: logical; if TRUE the old file is overwritten, a new file is written otherwise
- **edit.tag**: if the file is edited a different tag from the default ‘-edited’ can be used.
- **verbose**: whether to print information about successful edit

**Details**

The variables specified by parm within the .apsim file specified by file in the source directory src.dir are edited. The old values are replaced with value, which is a list that has the same number of elements as the length of the vector parm. The current .xml file will be overwritten if overwrite is set to TRUE; otherwise the file ‘file’ -edited.xml will be created. If (verbose = TRUE) then the name of the written file is returned. The function is similar to the edit_sim_file function in the ‘apsimr’ package, but with the difference that here the xml2 package is used instead.

**Value**

(when verbose=TRUE) complete file path to edited .xml file is returned as a character string. As a side effect this function creates a new XML file.

**Note**

This function cannot check whether replacement is of the correct length. Also, there is an inspect equivalent. It is more flexible than ‘edit_apsim’ and (perhaps) similar to ‘apsimr::edit_sim_file’.

**Examples**

```r
## This example changes the RUE values
extd.dir <- system.file("extdata", package = "apsimx")
values <- paste(rep(1.7, 12), collapse = " ")
## Writing to a temp directory, but replace as needed
tmp.dir <- tempdir()
edit_apsim_xml("Maize75.xml",
               src.dir = extd.dir,
               wrt.dir = tmp.dir,
               parm.path = ".//Model/rue",
               value = values)
```
**extract_values_apsimx**  
*Extract values from a parameter path*

**Description**

Extract initial values from a parameter path

**Usage**

```r
extract_values_apsimx(file, src.dir, parm.path)
```

**Arguments**

- `file`: file name to be run (the extension .apsimx is optional)
- `src.dir`: directory containing the .apsimx file to be run (defaults to the current directory)
- `parm.path`: parameter path either use inspect_apsimx or see example below

**Value**

a vector with extracted parameter values from an APSIM file.

**Examples**

```r
## Find examples
extd.dir <- system.file("extdata", package = "apsimx")
## Extract parameter path
pp <- inspect_apsimx("Maize.apsimx", src.dir = extd.dir,
                     node = "Manager", parm = list("Fert", 1))
ppa <- paste0(pp, ".Amount")
## Extract value
extract_values_apsimx("Maize.apsimx", src.dir = extd.dir, parm.path = ppa)
```

---

**get_apsimx_json**  
*fetches the json file for a specific model from APSIMX github*

**Description**

Retrieves the json replacement file for a specific model

**Usage**

```r
get_apsimx_json(model = "Wheat", wrt.dir = ".", cleanup = FALSE)
```
get_daymet2_apsim_met

Arguments

model a model (e.g. ‘Wheat’ or ‘Maize’)
wrt.dir directory to save the JSON file (default is the current directory)
cleanup whether to delete the JSON file

Details

Get APSIM-X Model Replacement from github

Value

a list read through the jsonlite package

See Also

insert_replacement_node

Examples

tmp.dir <- tempdir()
wheat <- get_apsimx_json(model = "Wheat", wrt.dir = tmp.dir)

get_daymet2_apsim_met

Get DAYMET data for an APSIM met file

Description

Uses download_daymet from the daymetr package to download data to create an APSIM met file.

Usage

get_daymet2_apsim_met(lonlat, years, wrt.dir = ".", filename, silent = FALSE)

Arguments

lonlat Longitude and latitude vector
years a numeric vector of years to extract
wrt.dir write directory (default is the current directory)
filename file name for writing out to disk
silent argument passed to download_daymet
get_daymet_apsim_met

Details

This function requires the `daymetr` package. This function should replace the `get_daymet_apsim_met` function.

If the filename is not provided it will not write the file to disk, but it will return an object of class 'met'. This is useful in case manipulation is required before writing to disk. The variable 'srad' as downloaded from daymet is average solar radiation, so it is converted to total. Daily total radiation (MJ/m2/day) can be calculated as follows: `((srad (W/m2) * dayl (s/day))/1,000,000)`

Vapor Pressure Deficit (vp) should be in hecto Pascals

Value

It returns an object of class 'met' and writes a file to disk when filename is supplied.

Source

The data is retrieved using the `daymetr` package. For the original source see: https://daymet.ornl.gov/

Examples

```r
## Not run:
require(daymetr)
## I write to a temp directory but replace as needed
dmet12 <- get_daymet2_apsim_met(lonlat = c(-93,42), years = 2012)
summary(dmet12)
## Check for reasonable ranges
check_apsim_met(dmet12)
## End(Not run)
```

get_daymet_apsim_met

Get DAYMET data for an APSIM met file

Description

Uses `get_daymet` from the `FedData` package to download data to create an APSIM met file.

Usage

```r
get_daymet_apsim_met(
  lonlat,
  years,
  wrt.dir = ".",
  filename = NULL,
  width.height = c(0.1 * 1.263012, 0.1),
  template,
  label = NULL,
)```
```
get_daymet_apsim_met

   elements = c("dayl", "prcp", "srad", "swe", "tmax", "tmin", "vp"),
   region = "na",
   tempo = "day",
   extraction.dir = paste0(tempdir(), "/FedData/extractions/daymet/", label, "/"),
   force.redo = FALSE,
   cleanup = FALSE
)

Arguments

lonlat    Longitude and latitude vector
years     a numeric vector of years to extract
wrt.dir   write directory
filename  file name for writing out to disk
width.height width and height of the cropped area (default 0.001, 0.001)
template  A Raster or Spatial object to serve as a template for cropping (see get_daymet).
label     a character string naming the area (see get_daymet)
elements  see get_daymet
region    see get_daymet
tempo     see get_daymet
extraction.dir see get_daymet
force.redo whether to delete download directories (default is FALSE). If the intention is for cleanup to delete all the files, ‘raw.dir’ and ‘extraction.dir’ should be supplied, supplying a sinlge name, such as ‘RAW’ and ‘EXTRACTION’.
cleanup   whether to delete download directories (default is FALSE). If the intention is for cleanup to delete all the files, ‘raw.dir’ and ‘extraction.dir’ should be supplied, supplying a sinlge name, such as ‘RAW’ and ‘EXTRACTION’.

Details

This function requires the FedData package.

If the filename is not provided it will not write the file to disk, but it will return an object of class ‘met’. This is useful in case manipulation is required before writing to disk. The variable ‘srad’ as downloaded from daymet is average solar radiation, so it is converted to total. Daily total radiation (MJ/m2/day) can be calculated as follows: ((srad (W/m2) * dayl (s/day)) / 1,000,000)
Vapor Pressure Deficit (vp) should be in hecto Pascals

Value

It returns an object of class ‘met’ and writes a file to disk when filename is supplied.

Source

The data is retrieved using the FedData package. For the original source see: https://daymet.ornl.gov/
```
## Examples

```r
## Not run:
require(FedData)
## I write to a temp directory but replace as needed
tmp.dir <- tempdir()
dmet12 <- get_daymet_apsim_met(lonlat = c(-93,42),
                               extraction.dir = paste0(tmp.dir,"/FedData/extractions/daymet/"),
                               years = 2012)
summary(dmet12)
## Check for reasonable ranges
check_apsim_met(dmet12)
## End(Not run)
```

---

**get_gsod_apsim_met**

Get GSOD data for an APSIM met file

### Description

Uses `get_GSOD` from the `GSODR` package to download data to create an APSIM met file.

### Usage

```r
get_gsod_apsim_met(
  lonlat,
  dates,
  wrt.dir = ".",
  filename = NULL,
  distance = 100,
  fillin.radn = FALSE
)
```

### Arguments

- `lonlat`: Longitude and latitude vector
- `dates`: Date ranges
- `wrt.dir`: Write directory
- `filename`: File name for writing out to disk
- `distance`: Distance in kilometers for the nearest station
- `fillin.radn`: Whether to fill in radiation data using the nasapower package. Default is `FALSE`.

### Details

This function requires the `GSODR` package.

If the filename is not provided it will not write the file to disk, but it will return an object of class `met`. This is useful in case manipulation is required before writing to disk.
get_iemre_apsim_met

Value

returns an object of class ‘met’ and writes a file to disk when filename is supplied.

Note

This source of data does not provide solar radiation. If ‘fillin.radn’ is TRUE it fill in radiation data using the nasapower package.

Examples

## Not run:
require(GSODR)
## This will not write a file to disk
gsd <- get_gsod_apsim_met(lonlat = c(-93,42), dates = c("2012-01-01","2012-12-31"))
summary(gsd)
## Check for reasonable ranges
## Radiation is not included by default
check_apsim_met(gsd)
## End(Not run)

get_iemre_apsim_met

Get weather data from Iowa Environmental Mesonet Reanalysis

Description

Retrieves weather data from Iowa Environmental Mesonet Reanalysis into an APSIM met file

Usage

get_iemre_apsim_met(
  lonlat,
  dates,
  wrt.dir = ".",
  filename = NULL,
  fillin.radn = FALSE
)

Arguments

lonlat Longitude and latitude vector
dates date ranges
wrt.dir write directory
filename file name for writing out to disk
fillin.radn whether to fill in radiation data using the nasapower package. Default is FALSE.
get_iem_apsim_met

Details

The original data can be obtained from: https://mesonet.agron.iastate.edu/iemre/

If the filename is not provided it will not write the file to disk, but it will return an object of class ‘met’. This is useful in case manipulation is required before writing to disk.

Value

returns an object of class ‘met’ and writes a file to disk when filename is supplied.

Note

Multi-year query is not supported for this product.

Examples

```r
## Not run:
## This will not write a file to disk
iemre <- get_iemre_apsim_met(lonlat = c(-93,42), dates = c("2012-01-01","2012-12-31"))
## Note that solar radiation is not available, but can be filled in
## using the nasapower package
iemre2 <- get_iemre_apsim_met(lonlat = c(-93,42),
                               dates = c("2012-01-01","2012-12-31"),
                               fillin.radn = TRUE)
summary(iemre)
summary(iemre2)

## Still it is important to check this object
## Since there is one day with missing solar radiation
check_apsim_met(iemre2)

## End(Not run)
```

get_iem_apsim_met

Get weather data from Iowa Environmental Ag Weather Stations

Description

Retrieves weather data from Iowa Environmental Mesonet (AgWeather) into an APSIM met file

Usage

```
get_iem_apsim_met(lonlat, dates, wrt.dir = ".", state, station, filename)
```
get_isric_soil_profile

Generate a synthetic APSIM soil profile from the ISRIC soil database

Description

Retrieves soil data from the ISRIC global database and converts it to an APSIM soil_profile object.
get_isric_soil_profile

Usage

get_isric_soil_profile(
  lonlat,
  statistic = c("mean", "Q0.5"),
  soil.profile,
  find.location.name = TRUE
)

Arguments

lonlat Longitude and latitude vector (e.g. c(-93, 42)).
statistic default is the mean
soil.profile a soil profile to fill in in case the default one is not appropriate
find.location.name default is TRUE. Use either maps package or photon API to find Country/State.
If you are running this function many times it might be better to set this to FALSE.

Details

Source: https://www.isric.org/
Details: https://www.isric.org/explore/soilgrids/faq-soilgrids


TODO: need to look into how this is done in APSIM NG https://github.com/APSIMInitiative/ApsimX/pull/3994/files

NOTE: Eric Zurcher provided help by sending me an R file originally written by Andrew Moore. It provides a bit of context for how some of the decisions were made for constructing the synthetic soil profiles in APSIM. (email from June 3 2021).

Variable which are directly retrieved and a simple unit conversion is performed:
* Bulk density - bdod
* Carbon - soc
* Clay - clay
* Sand - sand
* PH - phh2o
* Nitrogen - nitrogen

Variables which are estimated using pedotransfer functions:
LL15, DUL, SAT, KS, AirDry

TO-DO:
What do I do with nitrogen?
Can I use CEC?
How can I have a guess at FBiom and Finert?
FBiom does not depend on any soil property at the moment, should it?
get_power_apsim_met

Value

it generates an object of class 'soil_profile'.

Author(s)

Fernando E. Miguez, Eric Zurcher (CSIRO) and Andrew Moore (CSIRO)

See Also

apsimx_soil_profile, edit_apsim_replace_soil_profile, edit_apsimx_replace_soil_profile.

Examples

## Not run:
## Get soil profile properties for a single point
sp1 <- get_isric_soil_profile(lonlat = c(-93, 42))

## Visualize
plot(sp1)
plot(sp1, property = "water")

## End(Not run)

get_power_apsim_met  Get NASA-POWER data for an APSIM met file

Description

Uses get_power from the nasapower package to download data to create an APSIM met file.

Usage

get_power_apsim_met(lonlat, dates, wrt.dir = ".", filename = NULL)

Arguments

lonlat Longitude and latitude vector
dates date ranges
wrt.dir write directory
filename file name for writing out to disk

Details

This function requires the nasapower package.

If the filename is not provided it will not write the file to disk, but it will return an object of class 'met'. This is useful in case manipulation is required before writing to disk.
get_ssurgo_soil_profile

Value

returns an object of class ‘met’ and writes a file to disk when filename is supplied.

Examples

```r
## Not run:
require(nasapower)
## This will not write a file to disk
pwr <- get_power_apsim_met(lonlat = c(-93, 42), dates = c("2012-01-01","2012-12-31"))
## There are some missing values
summary(pwr)
## Check the met file
check_apsim_met(pwr)
## Impute using linear interpolation
pwr.imptd <- impute_apsim_met(pwr, verbose = TRUE)
summary(pwr.imptd)
## End(Not run)
```

get_ssurgo_soil_profile

Retrieve soil profile data and convert it to an object of class ‘soil_profile’

Description

Generate a synthetic soil profile based on the information in SSURGO database

Usage

```r
get_ssurgo_soil_profile(
  lonlat,
  shift = -1,
  nmapunit = 1,
  nsoil = 1,
  xout = NULL,
  soil.bottom = 200,
  method = c("constant", "linear"),
  nlayers = 10,
  verbose = FALSE
)
```

Arguments

- **lonlat**: Longitude and latitude vector (e.g. c(-93, 42))
- **shift**: simple mechanism for creating an area of interest by displacing the point indicated in lonlat by some amount of distance (e.g. 300 - in meters)
get_ssurgo_tables

retrieve soil profile data and return a table with data

Description

This function does partially what get_ssurgo_soil_profile does, but it returns a list with tables for mapunit, component, chorizon and mapunit.shp (object of class sf)

Usage

get_ssurgo_tables(lonlat, shift = -1, verbose = FALSE)
grep_json_list

Arguments

  lonlat  Longitude and latitude vector (e.g. c(-93, 42))
  shift  simple mechanism for creating an area of interest by displacing the point indicated in lonlat by some amount of distance (e.g. 300 - in meters)
  verbose  whether to print messages and warnings to the console default FALSE

Details

  Data source is USDA-NRCS Soil Data Access. See package soilDB for more details

Value

  a list with elements: mapunit, component, chorizon and mapunit.shp

Examples

  ## Not run:
  require(soilDB)
  require(sp)
  require(sf)
  require(spData)
  ## retrieve data from lon -93, lat = 42
  stbls <- get_ssurgo_tables(lonlat = c(-93, 42))

  ## End(Not run)

grep_json_list

Description

  grep but for json list

Usage

  grep_json_list(pattern, x, ignore.case = FALSE, search.depth = 10)

Arguments

  pattern  as in grep
  x  object (a list)
  ignore.case  as in grep
  search.depth  search depth for the list (to prevent endless search)

Value

  It returns a list with the found object, the json path and the positions in the list.
impute_apsim_met

Perform imputation for missing data in a met file

Description

Takes in an object of class ‘met’ and imputes values

Usage

impute_apsim_met(
  met,
  method = c("approx", "spline", "mean"),
  verbose = FALSE,
  ...
)

Arguments

met object of class ‘met’
method method for imputation, ‘approx’ (approxfun), ‘spline’ (splinefun) or ‘mean’ (mean).
verbose whether to print missing data to the console, default = FALSE
... additional arguments to be passed to imputation method

Value

an object of class ‘met’ with attributes

insert_replacement_node

Inserts a replacement node in a simple apsimx simulation file

Description

Inserts a replacement node in a simple apsimx simulation file

Usage

insert_replacement_node(
  file,
  src.dir,
  wrt.dir,
  rep.node,
  rep.node.position = 1,
  new.core.position = rep.node.position + 1,
insert_replacement_node

edit.tag = "-edited",
overwrite = FALSE,
verbose = TRUE,
root
)

Arguments

file file ending in .apsimx to be edited (JSON)
src.dir directory containing the .apsimx file to be edited; defaults to the current working
directory
wrt.dir should be used if the destination directory is different from the src.dir
rep.node replacement node as obtained by the get_apsimx_json function
rep.node.position position where the replacement node will be inserted, default is 1
new.core.position this by default will place the core simulation below the replacement node posi-
tion. With this option, this can be modified.
edit.tag if the file is edited a different tag from the default '-edited' can be used.
overwrite logical; if TRUE the old file is overwritten, a new file is written otherwise
verbose whether to print information about successful edit
root supply the node position in the case of multiple simulations such as factorials.

Value

it does not return an R object but it writes an apsimx file to disk

Examples

```r
## Not run:
tmp.dir <- tempdir()
wheat <- get_apsimx_json(model = "Wheat", wrt.dir = tmp.dir)
extd.dir <- system.file("extdata", package = "apsimx")
insert_replacement_node("Wheat.apsimx",
  src.dir = extd.dir, wrt.dir = tmp.dir,
  rep.node = wheat)

## End(Not run)
```
inspect_apsim

Inspect an .apsim (XML) file

Description

inspect an XML apsim file. It does not replace the GUI, but it can save time by quickly checking parameters and values.

Usage

inspect_apsim(
  file = "", 
  src.dir = ".", 
           "Outputfile", "Other"), 
                "InitialWater", "Sample", "SWIM"), 
  parm = NULL, 
  digits = 3, 
  print.path = FALSE, 
  root
)

Arguments

file    file ending in .apsim (Classic) to be inspected (XML)
src.dir directory containing the .apsim file to be inspected; defaults to the current working directory
soil.child specific soil component to be inspected
parm parameter to inspect when node = ‘Crop’, ‘Manager’, ‘Outputfile’ or ‘Other’
digits number of decimals to print (default 3)
print.path whether to print the parameter path (default = FALSE)
root root node label. In simulation structures such as factorials there will be multiple possible nodes. This can be specified by supplying an appropriate character.

Details

This is simply a script that prints the relevant parameters which are likely to need editing. It does not print all information from an .apsim file. For ‘Crop’, ‘Manager’ and ‘Other’, ‘parm’ should be indicated with a first element to look for and a second with the relative position in case there are multiple results.
Value

It returns the parameter path (when print.path equals TRUE) and table with inspected parameters and values.

Note

When multiple folders are present as it is the case when there are factorials. Inspect will find the instance in the first folder unless 'root' is supplied. By providing the name of the folder to root (or a regular expression), the appropriate node can be selected. In this case the printed path will be absolute instead of relative.

Examples

extd.dir <- system.file("extdata", package = "apsimx")
## Testing using 'Millet'
inspect_apsim("Millet.apsim", src.dir = extd.dir, node = "Clock")
inspect_apsim("Millet.apsim", src.dir = extd.dir, node = "Weather")
inspect_apsim("Millet.apsim", src.dir = extd.dir, node = "Soil", soil.child = "Metadata")
inspect_apsim("Millet.apsim", src.dir = extd.dir, node = "Soil", soil.child = "OrganicMatter")
inspect_apsim("Millet.apsim", src.dir = extd.dir, node = "Soil", soil.child = "Analysis")
inspect_apsim("Millet.apsim", src.dir = extd.dir, node = "Soil", soil.child = "InitialWater")
inspect_apsim("Millet.apsim", src.dir = extd.dir, node = "Soil", soil.child = "Sample")
inspect_apsim("Millet.apsim", src.dir = extd.dir, node = "Crop", parm = list("sow", NA))
inspect_apsim("Millet.apsim", src.dir = extd.dir, node = "Crop", parm = list("sow", 7))

## when soil.child = "Water" there are potentially many crops to chose from
## This selects LL, KL and XF for Barley
inspect_apsim("Millet.apsim", src.dir = extd.dir, node = "Soil", soil.child = "Water", parm = "Barley")
## This selects LL for all the crops
inspect_apsim("Millet.apsim", src.dir = extd.dir, node = "Soil", soil.child = "Water", parm = "LL")
## To print the parm.path the selection needs to be unique
## but still there will be multiple soil layers
## 'parm' can be a list or a character vector of length equal to two
inspect_apsim("Millet.apsim", src.dir = extd.dir, node = "Soil", soil.child = "Water", parm = list("Barley", "LL"), print.path = TRUE)

## Inspect outputfile
inspect_apsim("Millet.apsim", src.dir = extd.dir, node = "Outputfile", parm = "filename")
inspect_apsim("Millet.apsim", src.dir = extd.dir, node = "Outputfile", parm = "variables")

## Testing with maize-soybean-rotation.apsim
inspect_apsim("maize-soybean-rotation.apsim", src.dir = extd.dir, node = "Clock")
inspect_apsim("maize-soybean-rotation.apsim", src.dir = extd.dir, node = "Weather")
inspect_apsim("maize-soybean-rotation.apsim", src.dir = extd.dir, node = "Soil", soil.child = "Sample")
soil.child = "Metadata")
inspect_apsim("maize-soybean-rotation.apsim", src.dir = extd.dir, node = "Soil",
soil.child = "OrganicMatter")
inspect_apsim("maize-soybean-rotation.apsim", src.dir = extd.dir, node = "Soil",
soil.child = "Analysis")
inspect_apsim("maize-soybean-rotation.apsim", src.dir = extd.dir, node = "Soil",
soil.child = "InitialWater")
inspect_apsim("maize-soybean-rotation.apsim", src.dir = extd.dir, node = "Soil",
soil.child = "Sample")
inspect_apsim("maize-soybean-rotation.apsim", src.dir = extd.dir,
node = "SurfaceOrganicMatter")
inspect_apsim("maize-soybean-rotation.apsim", src.dir = extd.dir, node = "Crop")
## This has many options and a complex structure
## It is possible to select unique managements, but not non-unique ones
## The first element in parm can be a regular expression
inspect_apsim("maize-soybean-rotation.apsim", src.dir = extd.dir,
node = "Manager", parm = list("rotat",NA))
inspect_apsim("maize-soybean-rotation.apsim", src.dir = extd.dir,
node = "Manager",
parm = list("sow on a fixed date - maize",NA))
## Select an individual row by position
inspect_apsim("maize-soybean-rotation.apsim", src.dir = extd.dir,
node = "Manager",
parm = list("sow on a fixed date - maize",7))

## Illustrating the 'print.path' feature.
inspect_apsim("Millet.apsim", src.dir = extd.dir,
node = "Soil", soil.child = "Water",
parm = "DUL", print.path = TRUE)
## But the path can also be returned as a string
## Which is useful for later editing
pp <- inspect_apsim("Millet.apsim", src.dir = extd.dir,
node = "Soil", soil.child = "Water",
parm = "DUL", print.path = TRUE)

## Inspecting a factorial
## (Or simply a simulation with multiple folders)
## No cover
inspect_apsim("maize-factorial.apsim", src.dir = extd.dir,
root = "IA-CC_Canisteo_No-Cover")

## Cover
inspect_apsim("maize-factorial.apsim", src.dir = extd.dir,
root = "IA-CC_Canisteo_Cover")

inspect_apsimx  Inspect an .apsimx (JSON) file
**inspect_apsimx**

**Description**

inspect a JSON apsimx file. It does not replace the GUI, but it can save time by quickly checking parameters and values.

**Usage**

inspect_apsimx(
  file = "",
  src.dir = ".",
  parm = NULL,
  digits = 3,
  print.path = FALSE,
  root
)

**Arguments**

- **file**  
  file ending in .apsimx to be inspected (JSON)
- **src.dir**  
  directory containing the .apsimx file to be inspected; defaults to the current working directory
- **node**  
- **soil.child**  
  specific soil component to be inspected. The options vary depending on what is available (see details)
- **parm**  
  parameter to refine the inspection of the ‘manager’ list(‘parm’,’position’), use ‘NA’ for all the positions. ‘parm’ can be a regular expression for partial matching.
- **digits**  
  number of decimals to print (default 3). Not used now because everything is a character.
- **print.path**  
  whether to print the path to the specific parameter. Useful to give the later editing. (Also returned as ‘invisible’)
- **root**  
  root node label. In simulation structures such as factorials there will be multiple possible nodes. This can be specified by supplying an appropriate character.

**Details**

This is simply a script that prints the relevant parameters which are likely to need editing. It does not print all information from an .apsimx file. To investigate the available ‘soil.childs’ specify ‘Soil’ for ‘node’ and do not specify the ‘soil.child’.

**Value**

prints a table with inspected parameters and values (and ‘parm path’ when ‘print.path’ = TRUE).
Examples

```r
extd.dir <- system.file("extdata", package = "apsimx")
inspect_apsimx("Wheat.apsimx", src.dir = extd.dir, node = "Clock")
inspect_apsimx("Wheat.apsimx", src.dir = extd.dir, node = "Weather")
inspect_apsimx("Wheat.apsimx", src.dir = extd.dir, node = "Soil", soil.child = "Metadata")
inspect_apsimx("Wheat.apsimx", src.dir = extd.dir, node = "Soil", soil.child = "Physical")
inspect_apsimx("Wheat.apsimx", src.dir = extd.dir, node = "Soil", soil.child = "SoilWater")
inspect_apsimx("Wheat.apsimx", src.dir = extd.dir, node = "Soil", soil.child = "Organic")
inspect_apsimx("Wheat.apsimx", src.dir = extd.dir, node = "Soil", soil.child = "Chemical")
inspect_apsimx("Wheat.apsimx", src.dir = extd.dir, node = "Soil", soil.child = "InitialWater")
inspect_apsimx("Wheat.apsimx", src.dir = extd.dir, node = "Soil", soil.child = "InitialN")
inspect_apsimx("Wheat.apsimx", src.dir = extd.dir, node = "SurfaceOrganicMatter")
inspect_apsimx("Wheat.apsimx", src.dir = extd.dir, node = "MicroClimate")
inspect_apsimx("Wheat.apsimx", src.dir = extd.dir, node = "Crop")
inspect_apsimx("Wheat.apsimx", src.dir = extd.dir, node = "Manager")
inspect_apsimx("Wheat.apsimx", src.dir = extd.dir, node = "Report")

## Manager folder present
extd.dir <- system.file("extdata", package = "apsimx")
inspect_apsimx("maize-manager-folder.apsimx", node = "Other", src.dir = extd.dir, parm = list("Manager", "Fertiliser", "Amount"))
```

---

`inspect_apsimx_json`  
*Inspect an*.apsimx or .json (JSON) file

Description

inspect an .apsimx or .json (JSON) file. It does not replace the GUI, but it can save time by quickly checking parameters and values.

Usage

```r
inspect_apsimx_json(
  file = "",  
  src.dir = ".",  
  parm,  
  search.depth = 15,  
  print.path = FALSE,  
  verbose = FALSE
)
```

Arguments

- **file**: file ending in .apsimx or .json to be inspected (JSON)
**inspect_apsimx_replacement**

Inspect a replacement component in an .apsimx (JSON) file

**Description**

inspect the replacement component of an JSON apsimx file. It does not replace the GUI, but it can save time by quickly checking parameters and values.

**Parameters**

- `src.dir` directory containing the .apsimx or .json file to be inspected; defaults to the current working directory
- `parm` string or regular expression for partial matching.
- `search.depth` default is 15. How deep should the algorithm explore the structure of the list.
- `print.path` whether to print the parameter path (default is FALSE)
- `verbose` whether to print additional information (mostly used for debugging)

**Details**

This function is a work in progress. There are many instances for which it will not work.

It will probably only find the first instance that matches.

A future feature will be to search for a jspath instead of simply a regular expression

**Value**

prints a table with inspected parameters and values (and the path when ‘print.path’ = TRUE).

**Examples**

```r
extd.dir <- system.file("extdata", package = "apsimx")
## It seems to work for simple search
inspect_apsimx_json("Wheat.apsimx", src.dir = extd.dir, parm = "Version")
inspect_apsimx_json("Wheat.apsimx", src.dir = extd.dir, parm = "Simulations")
inspect_apsimx_json("Wheat.apsimx", src.dir = extd.dir, parm = "Clock")
inspect_apsimx_json("Wheat.apsimx", src.dir = extd.dir, parm = "Weather")
## Does return soil components
inspect_apsimx_json("Wheat.apsimx", src.dir = extd.dir, parm = "DUL")
## Or cultivar
inspect_apsimx_json("Wheat.apsimx", src.dir = extd.dir, parm = "Hartog")
```
Usage

inspect_apsimx_replacement(
  file = "",
  src.dir = ".",
  node = NULL,
  node.child = NULL,
  node.subchild = NULL,
  node.subsubchild = NULL,
  node.sub3child = NULL,
  node.sub4child = NULL,
  node.sub5child = NULL,
  node.string = NULL,
  root = list("Models.Core.Replacements", NA),
  parm = NULL,
  display.available = FALSE,
  digits = 3,
  print.path = FALSE,
  verbose = TRUE,
  grep.options
)

Arguments

file file ending in .apsimx to be inspected (JSON)
src.dir directory containing the .apsimx file to be inspected; defaults to the current working directory
node specific node to be inspected
node.child specific node child component to be inspected.
node.subchild specific node sub-child to be inspected.
node.subsubchild specific node sub-subchild to be inspected.
node.sub3child specific node sub3child to be inspected.
node.sub4child specific node sub4child to be inspected.
node.sub5child specific node sub5child to be inspected.
node.string passing of a string instead of the node hierarchy. Do not use this and also the other node arguments. This argument will overwrite the other node specifications.
root 'root' for the inspection of a replacement file (it gives flexibility to inspect other types of files).
parm specific parameter to display
display.available logical. Whether to display available components to be inspected (default = FALSE)
digits number of decimals to print (default 3)
inspect_apsim_xml

print.path  print the path to the inspected parameter (default FALSE)
verbose     whether to print additional information, default: TRUE
grep.options Additional options for grep. To be passed as a list.

Details
This is simply a script that prints the relevant parameters which are likely to need editing. It does not print all information from an .apsimx file.

Value
table with inspected parameters and values (and ‘parm path’ when ‘print.path’ = TRUE).

Note
I need to make some changes in order to be able to handle multiple parameters. At this point, it might work but it will generate warnings.

Examples

extd.dir <- system.file("extdata", package = "apsimx")
inspect_apsimx_replacement("MaizeSoybean.apsimx", src.dir = extd.dir,
    node = "Maize", node.child = "Phenology",
    node.subchild = "ThermalTime",
    node.subsubchild = "BaseThermalTime",
    node.sub3child = "Response")

## For Wheat
## getting down to 'XYPairs'
inspect_apsimx_replacement("WheatRye.apsimx",
    src.dir = extd.dir,
    node = "Wheat",
    node.child = "Structure",
    node.subchild = "BranchingRate",
    node.subsubchild = "PotentialBranchingRate",
    node.sub3child = "Vegetative",
    node.sub4child = "PotentialBranchingRate",
    node.sub5child = "XYPairs")

inspect_apsim_xml  Inspect an APSIM Classic auxiliary (XML) file

Description
inspect an auxiliary XML apsim file.
Usage

inspect_apsim_xml(
  file = "",  
  src.dir = ".",  
  parm,  
  verbose = TRUE,  
  print.path = TRUE
)

Arguments

file file ending in .xml to be inspected.
src.dir directory containing the .xml file to be inspected; defaults to the current working directory
parm parameter to inspect.
verbose Whether to print to standard output
print.path Whether to print the parameter path

Value

it returns an absolute parameter path(s)

Note

the behavior has changed from previous versions (earlier than 1.977). Before, if more than match was found it would return an error. Now it returns a list with all possible matches. This can be useful when trying to find a parameter.

Examples

extd.dir <- system.file("extdata", package = "apsimx")

inspect_apsim_xml("Maize75.xml", src.dir = extd.dir,  
  parm = "leaf_no_rate_change")

pp <- inspect_apsim_xml("Maize75.xml", src.dir = extd.dir,  
  parm = "leaf_no_rate_change",  
  verbose = FALSE,  
  print.path = FALSE)
mcmc.apsim.env

Environment to store data for apsim MCMC

Description
Environment which stores data for MCMC

Usage
mcmc.apsim.env

Format
An object of class environment of length 0.

Details
Create an apsim environment for MCMC

Value
This is an environment, so nothing to return.

mcmc.apsimx.env

Environment to store data for apsimx MCMC

Description
Environment which stores data for MCMC

Usage
mcmc.apsimx.env

Format
An object of class environment of length 0.

Details
Create an apsimx environment for MCMC

Value
This is an environment, so nothing to return.
napad_apsim_met

*Pad a met file with NAs when there are date discontinuities*

**Description**

It will fill in or ‘pad’ a met object with NAs

**Usage**

```r
napad_apsim_met(met)
```

**Arguments**

- `met` object of class ‘met’

**Details**

Fill in with missing data date discontinuities in a met file

**Value**

It returns an object of class ‘met’ with padded NAs.

**Note**

The purpose of this function is to allow for imputation using `impute_apsim_met`.

---

obsWheat

*Observed wheat phenology, LAI and biomass*

**Description**

Artificial observed data for Wheat

**Usage**

```r
obsWheat
```

**Format**

A data frame with 10 rows and 4 variables:

- **Date** -date- date starting Oct 1 2016 and ending June 6 2017
- **Wheat.Phenology.Stage** -numeric- phenology stage of wheat
- **Wheat.Leaf.LAI** -numeric- Leaf Area Index
- **Wheat.AboveGround.Wt** -numeric- above ground biomass (g/m²)
Details

A dataset containing the Date, phenology stage, LAI and above ground biomass for Wheat

Source

These are simulated data. For details see the APSIM documentation

optim_apsim

Optimize parameters in an APSIM simulation

Description

It is a wrapper for running APSIM and optimizing parameters using optim

Friendly printing of optim_apsim

Variance-Covariance for an ’optim_apsim’ object

Parameter estimates for an ’optim_apsim’ object

Usage

optim_apsim(
  file,
  src.dir = ".",
  crop.file,
  parm.paths,
  data,
  type = c("optim", "nloptr", "mcmc"),
  weights,
  index = "Date",
  parm.vector.index,
  xml.parm,
  ...
)

## S3 method for class 'optim_apsim'
print(x, ..., digits = 3, level = 0.95)

## S3 method for class 'optim_apsim'
vcov(object, ..., scaled = TRUE)

## S3 method for class 'optim_apsim'
coef(object, ..., scaled = FALSE)
Arguments

file  file name to be run (the extension .apsim is optional)
src.dir directory containing the .apsim file to be run (defaults to the current directory)
crop.file name of auxiliary xml file where parameters are stored. If this is missing, it is assumed that the parameters to be edited are in the main simulation file.
parm.paths absolute paths of the coefficients to be optimized. It is recommended that you use inspect_apsim or inspect_apsim_xml for this.
data data frame with the observed data. By default is assumes there is a 'Date' column for the index.
type Type of optimization. For now, optim and, if available, nloptr or 'mcmc' through runMCMC.
weights Weighting method or values for computing the residual sum of squares (see Note).
index Index for filtering APSIM output. 'Date' is currently used. (I have not tested how well it works using anything other than Date).
parm.vector.index Index to optimize a specific element of a parameter vector. At the moment it is possible to only edit one element at a time. This is because there is a conflict when generating multiple elements in the candidate vector for the same parameter.
xml.parm optional logical vector used when optimizing parameters which are both in the .apsim file and in the 'crop.file'. If 'crop.file' is missing it is assumed that the parameters to be optimized are in the .apsim file. If 'crop.file' is not missing it is assumed that they are in the 'crop.file'. If the parameters are in both, this needs to be specified in this argument.

... additional arguments (none used at the moment)
x object of class 'optim_apsim'
digits number of digits to round up the output
level confidence level (default 0.95)
object object of class 'optim_apsim'
scaled whether to return the scaled or unscaled estimates (TRUE in the optimized scale, FALSE in the original scale)

Details

Simple optimization for APSIM Classic

* This function assumes that you want to optimize parameters which are stored in an auxiliary XML file. These are typically crop or cultivar specific parameters. However, it is possible to optimize parameters present in the main simulation file.
* Only one observation per day is allowed in the data.
* Given how APSIM Classic works, this can only be run when the main simulation file is in the current directory and the crop file (or XML) should be in the same directory as the main simulation.
* The initial values for the optimization should be the ones in the stored crop parameter file.

* It is suggested that you keep a backup of the original file. This function will edit and overwrite the file during the optimization.

* When you use the parm.vector.index you cannot edit two separate elements of a vector at the same time. This should be used to target a single element of a vector only.

Value

object of class ‘optim_apsim’, but really just a list with results from optim and additional information.

prints to console

it returns the variance-covariance matrix for an object of class ‘optim_apsim’.

a numeric vector with the value of the parameter estimates.

Note

When computing the objective function (residual sum-of-squares) different variables are combined. It is common to weight them since they are in different units. If the argument weights is not supplied no weighting is applied. It can be ‘mean’, ‘var’ or a numeric vector of appropriate length.

This in the scale of the optimized parameters which are scaled to be around 1.

---

optim_apsimx

Optimize parameters in an APSIM Next Generation simulation

Description

It is a wrapper for running APSIM-X and optimizing parameters using optim

Usage

```r
optim_apsimx(
  file, 
  src.dir = ".", 
  parm.paths, 
  data, 
  type = c("optim", "nloptr", "mcmc"), 
  weights, 
  index = "Date", 
  parm.vector.index, 
  replacement, 
  root, 
  initial.values, 
  ...
)
```
Arguments

- **file**: file name to be run (the extension .apsimx is optional)
- **src.dir**: directory containing the .apsimx file to be run (defaults to the current directory)
- **parm.paths**: absolute or relative paths of the coefficients to be optimized. It is recommended that you use inspect_apsimx for this
- **data**: data frame with the observed data. By default assumes there is a ‘Date’ column for the index.
- **type**: Type of optimization. For now, optim and, if available, nloptr or ‘mcmc’ through runMCMC.
- **weights**: Weighting method or values for computing the residual sum of squares.
- **index**: Index for filtering APSIM output. Typically, “Date”, but it can be c(“report”, “Date”) for multiple simulations
- **parm.vector.index**: Index to optimize a specific element of a parameter vector. At the moment it is possible to only edit one element at a time. This is because there is a conflict when generating multiple elements in the candidate vector for the same parameter.
- **replacement**: TRUE or FALSE for each parameter. Indicating whether it is part of the ‘replacement’ component. Its length should be equal to the length of ‘parm.paths’.
- **root**: root argument for edit_apsimx_replacement
- **initial.values**: (required) supply the initial values of the parameters. (Working on fixing this...)

Details

Simple optimization for APSIM Next Generation

* At the moment it is required to provide starting values for the parameters of interest.
* It is suggested that you keep a backup of the original file. This function will edit and overwrite the file during the optimization.
* When you use the parm.vector.index you cannot edit two separate elements of a vector at the same time. This should be used to target a single element of a vector only. (I can add this feature in the future if it is justified.)

Value

object of class ‘optim_apsim’, but really just a list with results from optim and additional information.

Note

When computing the objective function (residual sum-of-squares) different variables are combined. It is common to weight them since they are in different units. If the argument weights is not supplied no weighting is applied. It can be ‘mean’, ‘variance’ or a numeric vector of appropriate length.
**Examples**

```r
## See the vignette for examples
```

---

**print.met**  
*Printer-friendly version of a metfile*

**Description**

Print a met file in a friendly way.

**Usage**

```r
## S3 method for class 'met'
print(x, ...)
```

**Arguments**

- `x`: an R object of class ‘met’
- `...`: additional printing arguments

**Value**

It prints to console. Not used to return an R object.

---

**read_apsim**  
*Read APSIM generated .out files*

**Description**

read ‘output’ databases created by APSIM runs (.out and .sim). One file at a time.

**Usage**

```r
read_apsim(
  file = "",
  src.dir = ".",
  value = c("report", "all"),
  date.format = "%d/%m/%Y"
)
```
Arguments

file  file name
src.dir  source directory where file is located
value  either ‘report’ (data.frame) or ‘all’ (list)
date.format  format for adding ‘Date’ column

Details

Read APSIM generated .out files

Value

This function returns a data frame with APSIM output or a list if value equals ‘all’

Examples

## Not run:
extd.dir <- system.file("extdata", package = "apsimx")
maize.out <- read_apsim("Maize", src.dir = extd.dir, value = "report")
millet.out <- read_apsim("Millet", src.dir = extd.dir, value = "report")

## End(Not run)

read_apsimx  Read APSIM-X generated .db files

Description

read SQLite databases created by APSIM-X runs. One file at a time.

Usage

read_apsimx(file = "", src.dir = ".", value = "report", simplify = TRUE)

Arguments

file  file name
src.dir  source directory where file is located
value  either ‘report’, ‘all’ (list) or user-defined for a specific report
simplify  if TRUE will attempt to simplify multiple reports into a single data.frame. If FALSE it will return a list.

Details

Read APSIM-X generated .db files
read_apsimx_all

Value

normally it returns a data frame, but it depends on the argument ‘value’ above

Note

if there is one single report it will return a data.frame. If there are multiple reports, it will attempt to merge them into a data frame. If not possible it will return a list with names corresponding to the table report names. It is also possible to select a specific report from several available by selecting ‘value = ReportName’, where ‘ReportName’ is the name of the specific report that should be returned. If you select ‘all’ it will return all the components in the data base also as a list.

read_apsimx_all  Read all APSIM-X generated .db files in a directory

Description

Like read_apsimx, but it reads all .db files in a directory.

Usage

read_apsimx_all(src.dir = ".", value = "report")

Arguments

src.dir  source directory where files are located
value    either ‘report’ or ‘all’ (only ‘report’ implemented at the moment)

Details

Read all APSIM-X generated .db files in a directory

Value

it returns a data frame or a list if ‘value’ equals ‘all’.

Note

Warning: very simple function at the moment, not optimized for memory or speed.
read_apsim_all  Read all APSIM generated .out files in a directory

Description

Like read_apsim, but it reads all .out files in a directory.

Usage

read_apsim_all(
  filenames,
  src.dir = ".",
  value = c("report", "all"),
  date.format = "%d/%m/%Y",
  simplify = TRUE
)

Arguments

filenames  names of files to be read
src.dir    source directory where files are located
value      either ‘report’ or ‘all’ (only ‘report’ implemented at the moment)
date.format format for adding ‘Date’ column
simplify   whether to return a single data frame or a list.

Details

Read all APSIM generated .out files in a directory

Value

returns a data frame or a list depending on the argument ‘simplify’ above.

Note

Warning: very simple function at the moment, not optimized for memory or speed.
Description

Read into R a met file and return an object of class ‘met’

Usage

read_apsim_met(file, src.dir = ".", verbose = TRUE)

Arguments

file path to met file
src.dir optional source directory
verbose whether to suppress all messages and warnings

Details

Read a met file into R

This function uses S3 classes and stores the additional information as attributes
I use a more strict format than APSIM and reading and writing will not
preserve all the details. For example, at this moment comments are lost through
the process of read and write unless they are added back in manually.
Also, empty lines are ignored so these will be lost as well in the read and write process.

Value

an object of class ‘met’ with attributes

Examples

extd.dir <- system.file("extdata", package = "apsimx")
ames.met <- read_apsim_met("Ames.met", src.dir = extd.dir)
ames.met
Sensitivity Analysis for APSIM Next Generation simulation

Description

It is a wrapper for running APSIM and evaluating different parameters values

Usage

sens_apsim(
  file,
  src.dir = ".",
  crop.file,
  parm.paths,
  parm.vector.index,
  xml.parm,
  grid,
  summary = c("mean", "max", "var", "sd", "none"),
  root,
  verbose = TRUE,
  
)

Arguments

file file name to be run (with extension .apsim)
src.dir directory containing the .apsim file to be run (defaults to the current directory)
crop.file name of auxiliary xml file where parameters are stored. If this is missing, it is assumed that the parameters to be edited are in the main simulation file.
parm.paths absolute or relative paths of the coefficients to be evaluated. It is recommended that you use inspect_apsim for this
parm.vector.index Index to evaluate a specific element of a parameter vector. At the moment it is possible to only edit one element at a time. This is because there is a conflict when generating multiple elements in the candidate vector for the same parameter.
xml.parm TRUE or FALSE for each parameter. Indicating whether it is part of an xml file. Its length should be equal to the length or 'parm.paths'.
grid grid of parameter values for the evaluation. It can be a data.frame.
summary function name to use to summarize the output to be a sinlge row (default is the mean).
root root argument for edit_apsim
verbose whether to print progress in percent and elapsed time.
... additional arguments (none used at the moment).
sens_apsimx

Value

object of class 'sens_apsim', but really just a list with results from the evaluations.

Note

The summary function is stored as an attribute of the data frame ‘grid.sims’.

Examples

## See the vignette for examples

---

sens_apsimx | Sensitivity Analysis for APSIM Next Generation simulation

Description

It is a wrapper for running APSIM-X and evaluating different parameters values

Summary computes variance-based sensitivity indexes from an object of class 'sens_apsim'

Usage

sens_apsimx(
  file,
  src.dir = ".",,
  parm.paths, 
  parm.vector.index, 
  replacement,
  grid, 
  summary = c("mean", "max", "var", "sd", "none"),
  root, 
  verbose = TRUE, 
  ...
)

## S3 method for class 'sens_apsim'
summary(object, ..., scale = FALSE, select = "all")

Arguments

file | file name to be run (the extension .apsimx is optional)
src.dir | directory containing the .apsimx file to be run (defaults to the current directory)
parm.paths | absolute or relative paths of the coefficients to be evaluated. It is recommended

that you use inspect_apsimx for this
soilwat_parms

parm.vector.index
Index to evaluate a specific element of a parameter vector. At the moment it is possible to only edit one element at a time. This is because there is a conflict when generating multiple elements in the candidate vector for the same parameter.

replacement
TRUE or FALSE for each parameter. Indicating whether it is part of the ‘replacement’ component. Its length should be equal to the length or ‘parm.paths’.

grid
grid of parameter values for the evaluation. It can be a data.frame.

summary
function name to use to summarize the output to be a single row (default is the mean).

root
root argument for edit_apsimx_replacement

verbose
whether to print progress in percent and elapsed time.

...
additional arguments (none used at the moment)

object
object of class ‘sens_apsim’

scale
if all inputs are numeric it is better to scale them. The default is FALSE as some inputs might be characters or factors. In this case all inputs will be treated as factors in the sum of squares decomposition.

select
option for selecting specific variables in the APSIM output. It will be treated as a regular expression

Value
object of class ‘sens_apsim’, but really just a list with results from the evaluations.

prints to console

Note
The summary function is stored as an attribute of the data frame ‘grid.sims’.

Examples

## See the vignette for examples

---

soilwat_parms

Helper function to supply SoilWat parameters

Description

Creates a list with specific components for the SoilWat model
soilwat_parms

Usage

soilwat_parms(
  SummerCona = NA,
  SummerU = NA,
  SummerDate = NA,
  WinterCona = NA,
  WinterU = NA,
  WinterDate = NA,
  DiffusConst = NA,
  DiffusSlope = NA,
  Salb = NA,
  CN2Bare = NA,
  CNRed = NA,
  CNCov = NA,
  Slope = NA,
  Discharge = NA,
  CatchmentArea = NA,
  MaxPond = NA,
  SWCON = NA
)

Arguments

SummerCona see APSIM documentation
SummerU see APSIM documentation
SummerDate see APSIM documentation
WinterCona see APSIM documentation
WinterU see APSIM documentation
WinterDate see APSIM documentation
DiffusConst see APSIM documentation
DiffusSlope see APSIM documentation
Salb soil albedo (see APSIM documentation)
CN2Bare see APSIM documentation
CNRed see APSIM documentation
CNCov see APSIM documentation
Slope see APSIM documentation
Discharge see APSIM documentation
CatchmentArea see APSIM documentation
MaxPond see APSIM documentation
SWCON see APSIM documentation

Details

current documentation for APSIM 7.10 https://www.apsim.info/documentation/model-documentation/soil-modules-documentation/soilwat/
ssurgo2sp

**Value**

a ‘list’ with class ‘soilwat_parms’

**Description**

Utility function to convert SSURGO data to soil profile

**Usage**

```r
ssurgo2sp(
  mapunit = NULL,
  component = NULL,
  chorizon = NULL,
  mapunit.shp = NULL,
  nmapunit = 1,
  nsoil = 1,
  xout = NULL,
  soil.bottom = 200,
  method = c("constant", "linear"),
  nlayers = 10,
  verbose = FALSE
)
```

**Arguments**

- **mapunit**: mapunit SSURGO file
- **component**: component SSURGO file
- **chorizon**: chorizon SSURGO file
- **mapunit.shp**: mapunit shapefile for creating metadata
- **nmapunit**: number of mapunits to select
- **nsoil**: number of soil components (within a mapunit) to consider
- **xout**: vector for interpolation and extrapolation
- **soil.bottom**: bottom of the soil profile
- **method**: method used for interpolation (see `approx`)
- **nlayers**: number of soil layers to generate
- **verbose**: whether to print details of the process

**Details**

Download the data from SSURGO using the ‘FedData’ package
This will generate csv files ‘chorizon’, ‘component’ and ‘mapunit’, but also many other files which are not needed for creating a soil profile.
Value

a list with soil profile matrices with length equal to nsoil

Examples

```r
require(ggplot2)
require(sf)
extd.dir <- system.file("extdata", package = "apsimx")

chorizon <- read.csv(paste0(extd.dir,"/ISUAG/SSURGO/ISUAG_SSURGO_chorizon.csv"))
component <- read.csv(paste0(extd.dir,"/ISUAG/SSURGO/ISUAG_SSURGO_component.csv"))
mapunit <- read.csv(paste0(extd.dir,"/ISUAG/SSURGO/ISUAG_SSURGO_mapunit.csv"))
mapunit.shp <- st_read(paste0(extd.dir,"/ISUAG/SSURGO/ISUAG_SSURGO_Mapunits.shp"), quiet = TRUE)

## Using default 'constant' method
sp.c <- ssurgo2sp(mapunit = mapunit,
                  component = component,
                  chorizon = chorizon,
                  mapunit.shp = mapunit.shp)

sp.c <- sp.c[[1]]

ggplot(data = sp.c, aes(y = -Depth, x = Carbon)) +
  geom_point() +
  geom_path() +
  ylab("Soil Depth (cm)") + xlab("Organic Matter (percent)") +
  ggtitle("method = constant")

## Using 'linear' method
sp.l <- ssurgo2sp(mapunit = mapunit,
                  component = component,
                  chorizon = chorizon,
                  mapunit.shp = mapunit.shp,
                  method = "linear")

sp.l <- sp.l[[1]]

ggplot(data = sp.l, aes(y = -Depth, x = Carbon)) +
  geom_point() +
  geom_path() +
  ylab("Soil Depth (cm)") + xlab("Organic Matter (percent)") +
  ggtitle("Method linear")

## Not run:
## Method using get_ssurgo_tables

require(soilDB)
require(sp)
require(sf)
```
**Description**

Creates a list with specific components for the SWIM model

**Usage**

```r
swim_parms(
  Salb = NA,
  CN2Bare = NA,
  CNRed = NA,
  CNCov = NA,
  KDul = NA,
  PSIDul = NA,
  VC = NA,
)```
swim_Parms

DTmin = NA,
DTmax = NA,
MaxWaterIncrement = NA,
SpaceWeightingFactor = NA,
SoluteSpaceWeightingFactor = NA,
Diagnostics = NA,
SwimWaterTable_WaterTableDepth = NA,
SwimSubsurfaceDrain_DrainDepth = NA,
SwimSubsurfaceDrain_DrainSpacing = NA,
SwimSubsurfaceDrain_DrainRadius = NA,
SwimSubsurfaceDrain_Klat = NA,
SwimSubsurfaceDrain_ImpermDepth = NA

Arguments

Salb                  see APSIM documentation
CN2Bare               see APSIM documentation
CNRed                 see APSIM documentation
CNCov                 see APSIM documentation
KDul                  see APSIM documentation
PSIDul                see APSIM documentation
VC                    see APSIM documentation
DTmin                 see APSIM documentation
DTmax                 see APSIM documentation
MaxWaterIncrement     see APSIM documentation
SpaceWeightingFactor  see APSIM documentation
SoluteSpaceWeightingFactor see APSIM documentation
Diagnostics           see APSIM documentation
SwimWaterTable_WaterTableDepth see APSIM documentation
SwimSubsurfaceDrain_DrainDepth see APSIM documentation
SwimSubsurfaceDrain_DrainSpacing see APSIM documentation
SwimSubsurfaceDrain_DrainRadius see APSIM documentation
SwimSubsurfaceDrain_Klat see APSIM documentation
SwimSubsurfaceDrain_ImpermDepth see APSIM documentation
Details

current documentation for APSIM 7.10 https://www.apsim.info/documentation/model-documentation/
  soil-modules-documentation/swim3/

Value

  a ‘list’ with class ‘swim_parms’


---

**tt_apsim_met**

*Calculates Thermal Time taking a ‘met’ object*

Description

Calculates Thermal Time using the ‘Classic’ formula, Heat Stress, Crop Heat Unit and other methods

Usage

```r
(tt_apsim_met(
  met, dates, method = c("Classic_TT", "HeatStress_TT", "CropHeatUnit_TT", "APSIM_TT", "CERES_TT", "all"),
  x_temp = c(0, 26, 34),
  y_tt = c(0, 26, 0),
  base_temp = 0,
  max_temp = 30,
  dates.format = c("%d-%m")
))
```

Arguments

- `met` object of class ‘met’
- `dates` when the calculation starts and when it ends. At the moment it needs to be a character vector (e.g. c('01-05', '10-10')). It will use the same dates every year for multiple years.
- `method` one of ‘Classic_TT’, ‘HeatStress_TT’, ‘APSIM_TT’, ‘CERES_TT’ and ‘all’
- `x_temp` cardinal temperatures (base, optimal and maximum)
- `y_tt` thermal time accumulation for cardinal temperatures
- `base_temp` base temperature for Classic TT calculation
- `max_temp` maximum temperature for Classic TT calculation
- `dates.format` default is ‘%d-%m’ which means day and month

Details

Calculating Thermal Time using a variety of methods
Value

it returns an object of class ‘met’ with additional columns ‘Date’ and the corresponding TT calculation

References


Examples

```r
## Not run:
require(nasapower)
require(ggplot2)
pwr <- get_power_apsim_met(lonlat = c(-93,42), dates = c("2012-01-01","2015-12-31"))
check_apsim_met(pwr)
pwr <- impute_apsim_met(pwr)
pwr2 <- tt_apsim_met(pwr, dates = c("01-05", "30-10"), method = c("Classic", "Heat"))
ggplot(data = pwr2, aes(x = Date, y = Classic_TT)) + geom_point()
ggplot(data = pwr2, aes(x = Date, y = HeatStress_TT)) + geom_point()

## End(Not run)
```

---

**unit_conv** performs common unit conversions

Description

It does not do much at the moment, but I will develop it as I have increasing needs for unit conversions.

Usage

```
unit_conv(x, from, to)
```

Arguments

- **x**: input variable
- **from**: original units
- **to**: target units
Details

Function which performs common unit conversions

At the moment possible conversions are:

- ‘g/m2’ to ‘kg/ha’
- ‘kg/ha’ to ‘g/m2’
- ‘lb’ to ‘kg’
- ‘kg’ to ‘lb’
- ‘maize bu’ to ‘kg’
- ‘kg’ to ‘maize bu’
- ‘soy bu’ to ‘kg’
- ‘kg’ to ‘soy bu’
- ‘maize bu/ac’ to ‘kg/ha’
- ‘maize bu/ac’ to ‘g/m2’
- ‘kg/ha’ to ‘maize bu/ac’
- ‘g/m2’ to ‘maize bu/ac’
- ‘soy bu/ac’ to ‘kg/ha’
- ‘soy bu/ac’ to ‘g/m2’
- ‘kg/ha’ to ‘soy bu/ac’
- ‘g/m2’ to ‘soy bu/ac’
- ‘mm’ to ‘inches’
- ‘inches’ to ‘mm’

This is for metric and Imperial conversions Source: https://www.extension.iastate.edu/agdm/wholefarm/html/c6-80.html

Value

value of the input variable with new units

Examples

grain.yield.gm2 <- 600
grain.yield.kgha <- unit_conv(grain.yield.gm2, from = “g/m2“, to = “kg/ha“)
grain.yield.kgha
Description

Generate an interactive viewer for an APSIM file

Usage

view_apsim(file, src.dir, viewer = c("json", "react"), ...)

Arguments

file  a file ending in .apsim to be inspected (XML)
src.dir directory containing the .apsim file to be inspected; defaults to the current working directory
viewer either “json” or “react”.
... additional arguments passed to either ‘jsonedit’ or ‘reactjson’. These are functions in package listviewer.

Value

a display with the APSIM file structure.

Note

I do not know how to edit an APSIM file using this method yet.

Examples

extd.dir <- system.file("extdata", package = "apsimx")
## View the structure of the APSIM-X simulation file
view_apsim("Millet.apsim", src.dir = extd.dir)
view_apsimx  Viewing an APSIM-X file interactively

Description

Generate an interactive viewer for an APSIM-X file

Usage

view_apsimx(file, src.dir, viewer = c("json", "react"), ...)

Arguments

file  a file ending in .apsimx to be inspected (JSON)
src.dir  directory containing the .apsimx file to be inspected; defaults to the current working directory
viewer  either “json” or “react”.
...  additional arguments passed to either ‘jsonedit’ or ‘reactjson’. These are functions in package listviewer.

Value

a display with the APSIM file structure.

Note

I do not know how to edit an APSIM-X file using this method yet.

Examples

extd.dir <- system.file("extdata", package = "apsimx")
## View the structure of the APSIM-X simulation file
view_apsimx("Wheat.apsimx", src.dir = extd.dir)
view_apsim_xml

View an APSIM Classic auxiliary (XML) file

Description

view an auxiliary XML apsim file.

Usage

view_apsim_xml(file, src.dir, viewer = c("json", "react"), ...)

Arguments

file          file ending in .xml to be viewed.
src.dir       directory containing the .xml file to be viewed; defaults to the current working directory
viewer        either “json” or “react”.
...            additional arguments passed to either ‘jsonedit’ or ‘reactjson’.

Details

view APSIM XML file

Value

It does not return an object but it produces a tree display of the APSIM file.

Examples

extd.dir <- system.file("extdata", package = "apsimx")
view_apsim_xml("Maize75.xml", src.dir = extd.dir)
### wop

**Wheat example optimization results**

**Description**
Results from Wheat optimization example

**Usage**

```r
wop
```

**Format**

An object of class `optim_apsim`

- `wop`  wheat optimization results

**Source**

Result of running the examples in Parameter Optimization vignette

---

### wop.h

**Wheat example optimization results plus Hessian**

**Description**

Results from Wheat optimization example plus the Hessian

**Usage**

```r
wop.h
```

**Format**

An object of class `optim_apsim`

- `wop.h`  wheat optimization results plus Hessian

**Source**

Result of running the examples in Parameter Optimization vignette with the added Hessian
write_apsim_met  Write an APSIM met file

Description

Write an object of class ‘met’ to disk

Usage

write_apsim_met(met, wrt.dir = NULL, filename = NULL)

Arguments

met  object of class ‘met’
wrt.dir  directory where the file will be written
filename  optional alternative filename

Details

Write a met file to disk. It takes an object of class ‘met’
at the moment the read-write cycle will strip comments

Value

does not create an R object, it only writes to disk

Examples

extd.dir <- system.file("extdata", package = "apsimx")
ames.met <- read_apsim_met("Ames.met", src.dir = extd.dir)
ames.met
tmp.dir <- tempdir()
write_apsim_met(ames.met, wrt.dir = tmp.dir, filename = "Ames.met")
## Here I write to a temporary directory, but change this to where
## you want to write to
xargs_apsimx

Provide extra arguments for APSIM-X

Description

This provides additional command line arguments when running the model.

Usage

```r
xargs_apsimx(
  verbose = FALSE,
  csv = FALSE,
  merge.db.files = FALSE,
  list.simulations = FALSE,
  list.referenced.filenames = FALSE,
  single.threaded = FALSE,
  cpu.count = -1L,
  simulation.names = FALSE
)
```

Arguments

- **verbose**: Write detailed messages to stdout when a simulation starts/finishes.
- **csv**: Export all reports to .csv files.
- **merge.db.files**: Merge multiple .db files into a single .db file.
- **list.simulations**: List simulation names without running them.
- **list.referenced.filenames**: List all files that are referenced by an .apsimx file(s).
- **single.threaded**: Run all simulations sequentially on a single thread.
- **cpu.count**: (Default: -1) Maximum number of threads/processes to spawn for running simulations.
- **simulation.names**: Only run simulations if their names match this regular expression.

Details

Extra arguments for running APSIM-X

Value

it returns a character vector with the extra arguments.
Index

* datasets
  * apsim.options, 4
  * apsimx.options, 6
  * mcmc.apsim.env, 59
  * mcmc.apsimx.env, 59
  * obsWheat, 60
  * wop, 84
  * wop.h, 84

  approx, 74
  approxfun, 48
  apsim, 3
  apsim.options, 4
  apsim_example, 13
  apsim_options, 14
  apsim_version, 15
  apsimx, 5
  apsimx.options, 6
  apsimx_example, 7
  apsimx_filetype, 8
  apsimx_options, 9
  apsimx_soil_profile, 10, 12, 44
  as_apsim_met, 15
  auto_detect_apsim_examples, 17
  auto_detect_apsimx_examples, 16
  check_apsim_met, 16, 18
  check_apsimx_soil_profile
    (apsimx_soil_profile), 10
  coef.optim_apsim (optim_apsim), 61
  compare_apsim, 18
  compare_apsim_met, 20
  date2doy (doy2date), 21
  download_daymet, 36
doy2date, 21

  edit_apsim, 22, 70
  edit_apsim_replace_soil_profile, 32, 44
  edit_apsim_xml, 33

  edit_apsimx, 24
  edit_apsimx_batch, 26
  edit_apsimx_replace_soil_profile, 30, 44
  edit_apsimx_replacement, 28, 64, 72
  extract_values_apsimx, 35
  get_apsimx_json, 35, 49
get_daymet, 37, 38
  get_daymet2_apsim_met, 36
  get_daymet_apsim_met, 37, 37
  get_GSOD, 39
  get_gsd_apsim_met, 39
  get_iem_apsim_met, 41
  get_iemre_apsim_met, 40
  get_isric_soil_profile, 42
  get_power, 44
  get_power_apsim_met, 44
  get_ssurgo_soil_profile, 45
  get_ssurgo_tables, 46
  grep_json_list, 47
  impute_apsim_met, 48, 60
insert_replacement_node, 36, 48
inspect_apsim, 50, 62, 70
  inspect_apsim_xml, 57, 62
inspect_apsimx, 11, 52, 64, 71
inspect_apsimx_json, 54
inspect_apsimx_replacement, 55

  mcmc.apsim.env, 59
  mcmc.apsimx.env, 59
  mean, 48

  napad_apsim_met, 60
  nloptr, 62, 64

  obsWheat, 60
  optim, 61–64
  optim_apsim, 61
  optim_apsimx, 63
plot.met_mrg(compare_apsim_met), 20
plot.out_mrg(compare_apsim), 18
plot.soil_profile
  (apsimx_soil_profile), 10
print.met, 65
print.optim_apsim(optim_apsim), 61

read_apsim, 65, 68
read_apsim_all, 68
read_apsim_met, 69
read_apsimx, 66, 67
read_apsimx_all, 67
runMCMC, 62, 64

sens_apsim, 70
sens_apsimx, 71
soilwat parms, 72
splinefun, 48
ssurgo2sp, 46, 74
summary.sens_apsim(sens_apsimx), 71
swim_parms, 76

tt_apsim_met, 78

unit_conv, 79

vcov.optim_apsim(optim_apsim), 61
view_apsim, 81
view_apsim_xml, 83
view_apsimx, 82

wop, 84
wop.h, 84
write_apsim_met, 85

xargs_apsimx, 86