Package 'rapsimng'

May 6, 2026

Type Package

Title APSIM Next Generation

Version 0.5.0

Description The Agricultural Production Systems sIMulator ('APSIM') is a widely used to simulate the agricultural systems for multiple crops. This package is designed to create, modify and run 'apsimx' files in the 'APSIM' Next Generation <https://www.apsim.info/>.

License MIT + file LICENSE

URL https://rapsimng.bangyou.me/, https://github.com/byzheng/rapsimng

BugReports https://github.com/byzheng/rapsimng/issues

Encoding UTF-8

Depends R (>= 4.1.0)

Imports jsonlite, tibble, magrittr, dplyr, purrr, rlang, DBI, RSQLite, tidyweather

Suggests testthat, knitr, rmarkdown, tidyverse, methods

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Bangyou Zheng [aut, cre]

Maintainer Bangyou Zheng <zheng.bangyou@gmail.com>

Repository CRAN

Date/Publication 2026-05-06 11:00:02 UTC

Contents

`append_model` *append a model into apsimx*

Description

append a model into apsimx

Usage

```
append_model(l, path, model)
```

Arguments

`l`	the list of apsimx file
`path`	If numeric, the path returned by search_path or search_node. If character, the path supported by apsimx
`model`	A new model which should be a list of new models

Value

The modified list with new value

Examples

```
wheat <- read_apsimx(system.file("extdata/wheat.apsimx", package = "rapsimng"))
replacements <- new_model("Core.Replacements")
wheat_new <- insert_model(wheat, 1, replacements)
replacements_node <- search_path(wheat_new, ".Simulations.Replacements")
replacements_node$path
# Add a cultivar folder under replacements
cultivar_folder <- new_model("PMF.CultivarFolder", "Cultivars")
wheat_new <- insert_model(wheat_new, replacements_node$path, cultivar_folder)
cultivar_folder_node <- search_path(wheat_new,
                                    ".Simulations.Replacements.Cultivars")
cultivar_folder_node$path
# Add an new cultivar
cultivar <- new_model("PMF.Cultivar", "Hartog")
wheat_new <- insert_model(wheat_new, cultivar_folder_node$path, cultivar)
cultivar_node <- search_path(wheat_new,
                             ".Simulations.Replacements.Cultivars.Hartog")
cultivar_node$path
# Append another cultivar
cultivar2 <- new_model("PMF.Cultivar", "Axe")
wheat_new <- append_model(wheat_new, cultivar_node$path, list(cultivar2))
cultivar2_node <- search_path(wheat_new,
                              ".Simulations.Replacements.Cultivars.Axe")
cultivar2_node$path
```

---

disable_models *Disable models in apsimx*

Description

Disable models in apsimx

Usage

```
disable_models(l, paths)
```

Arguments

l	the list of apsimx file
paths	If numeric, the path returned by search_path or search_node. If character, the path supported by apsimx

Value

The modified list with new value

Examples

```
wheat <- read_apsimx(system.file("extdata/Wheat.json", package = "rapsimng"))
a <- disable_models(wheat, '[Wheat].Phenology.ThermalTime')
```

`frost_heat_damage` *Frost and Heat Damage Functions*

Description

Helper and wrapper functions to calculate frost and heat damage from temperature and growth stage.

Usage

```
default_frost_heat_params()

stage_sens_frost(growth_stage, params)

stage_sens_heat(growth_stage, params)

temp_damage_frost(mint, params)

temp_damage_heat(maxt, params)

daily_damage_frost(mint, growth_stage, params)

daily_damage_heat(maxt, growth_stage, params)

daily_damage_frost_heat(mint, maxt, growth_stage, params)

cum_damage_frost(mint, growth_stage, params)

cum_damage_heat(maxt, growth_stage, params)

cum_damage_frost_heat(mint, maxt, growth_stage, params)

calc_daily_damage_frost_heat(
  mint,
  maxt,
  growth_stage,
  crop = c("Wheat", "Canola"),
  params = NULL
)

calc_cum_damage_frost_heat(
  mint,
  maxt,
```

```
  growth_stage,
  crop = c("Wheat", "Canola"),
  params = NULL
)
```

Arguments

growth_stage	Numeric vector of daily growth stage.
params	A crop parameter list. If provided, it will override the crop argument.
mint	Numeric vector of daily minimum temperature.
maxt	Numeric vector of daily maximum temperature.
crop	Crop name to use from default_frost_heat_params().

Details

The damage functions are based on the APSIM NG Model FrostHeatDamageFunctions. See full documentation here.

Value

A named list with crop-specific parameter lists.

A numeric vector of frost stage sensitivity (0-1).

A numeric vector of heat stage sensitivity (0-1).

A numeric vector of frost damage contribution from temperature.

A numeric vector of heat damage contribution from temperature.

A numeric vector of daily frost damage.

A numeric vector of daily heat damage.

A numeric vector of combined daily frost and heat damage.

A scalar cumulative frost damage.

A scalar cumulative heat damage.

A scalar cumulative frost and heat damage.

A tibble with daily stage sensitivity, temperature damage, and daily frost/heat/combined damage.

A tibble with cumulative frost damage, cumulative heat damage, and cumulative combined frost-heat damage.

Examples

```
default_frost_heat_params()
calc_daily_damage_frost_heat(
  mint = c(-2, 0, 4),
  maxt = c(29, 33, 36),
  growth_stage = c(7.0, 8.0, 9.0),
  crop = "Wheat"
)
calc_cum_damage_frost_heat(
```

```
  mint = c(-2, 0, 4),
  maxt = c(29, 33, 36),
  growth_stage = c(7.0, 8.0, 9.0),
  crop = "Wheat"
)
```

`get_cultivar` *Get all cultivar parameters in a model*

Description

Get all cultivar parameters in a model

Usage

```
get_cultivar(l, alias = TRUE)
```

Arguments

`l`	The list of apsimx file
`alias`	Whether to export alias

Value

A data frame for all cultivar parameters

Examples

```
wheat <- read_apsimx(system.file("extdata/Wheat.json", package = "rapsimng"))
get_cultivar(wheat)
get_cultivar(wheat, alias = FALSE)
```

`get_experiment` *Get an experiment node by name*

Description

Get an experiment node by name

Usage

```
get_experiment(apsimx, experiment)
```

Arguments

apsimx	A parsed APSIMX model from read_apsimx().
experiment	A single experiment name to locate.

Value

An apsimxNode object for the experiment.

Examples

```
wheat <- read_apsimx(system.file("extdata/wheat.apsimx", package = "rapsimng"))
exp <- search_path(wheat, path = "[Experiment]")
get_experiment(wheat, exp$node$Name)
```

---

get_metfile *Get the met file name for an experiment*

Description

Get the met file name for an experiment

Usage

```
get_metfile(l, is_stop = TRUE)
```

Arguments

l	A list or apsimxNode red by read_apsimx
is_stop	Whether stop the function when error

Value

The met file name in a experiment

Examples

```
wheat <- read_apsimx(system.file("extdata/wheat.apsimx", package = "rapsimng"))
exp <- search_path(wheat, path = "[Experiment]")
get_metfile(exp)
```

get_parent *Get the parent node from a path*

Description

Get the parent node from a path

Usage

```
get_parent(l, path)
```

Arguments

l	the list of apsimx file
path	If numeric, the path returned by search_path or search_node. If character, the path supported by apsimx

Value

A new list for parent

Examples

```
wheat <- read_apsimx(system.file("extdata/Wheat.json", package = "rapsimng"))
a <- search_path(wheat, '[Structure].BranchingRate')
get_parent(wheat, a$path)
```

get_pawc *Estimate Plant Available Water Capacity (PAWC) from an apsimx file*

Description

This function calculates the Plant Available Water Capacity (PAWC) from soil properties in an APSIM Next Generation (.apsimx) file. PAWC represents the amount of water that can be stored in the soil and is available for plant uptake.

Usage

```
get_pawc(l, soil_path = NULL, crop = NULL, depth = NULL)
```

Arguments

`l`	The list of apsimx file
`soil_path`	The path to the soil model. If NULL (default), searches for the first Soil model. Can be a character string specifying the soil name.
`crop`	The crop name for crop-specific lower limit (LL). If NULL (default), uses LL15 and does not apply XF. If provided, uses the crop-specific LL and XF values from the SoilCrop parameters.
`depth`	The depth (in mm) to which PAWC should be calculated. If NULL (default), calculates for all layers. If specified, applies proportional weighting to layers that are partially or completely beyond this depth.

Details

PAWC represents the soil's water storage capacity between field capacity (DUL) and the permanent wilting point (LL). It is a critical parameter for:

PAWC is calculated for each soil layer using the formula:

When crop = NULL and depth = NULL:

$$PAWC_i = (DUL_i - LL15_i) \times Thickness_i$$

When crop is specified:

$$PAWC_i = (DUL_i - LL_i) \times Thickness_i \times XF_i$$

When depth is specified (with or without crop):

$$PAWC_i = (DUL_i - LL_i) \times Thickness_i \times [XF_i] \times Weight_i$$

Where:

- **DUL** (Drained Upper Limit): The volumetric water content (mm/mm) after the soil has drained under gravity, typically 2-3 days after saturation
- **LL** (Lower Limit): The volumetric water content (mm/mm) below which plants cannot extract water. This can be either:
  - **LL15**: Generic lower limit at 15 bar suction (used when crop = NULL)
  - **Crop-specific LL**: Lower limit specific to a crop type, accounting for different rooting characteristics
- **Thickness**: Layer thickness in millimeters
- **XF**: Exploration factor (0-1), representing the fraction of a layer explored by roots. Only applied when crop is specified; represents the proportion of the layer volume that roots can explore
- **Weight**: Depth weighting factor (0-1), only applied when depth is specified:
  - 1.0 for layers completely within the specified depth
  - Proportional fraction for layers partially within the depth
  - 0.0 for layers completely beyond the specified depth

The total PAWC is the sum across all layers:

$$PAWC_{total} = \sum_{i=1}^{n} PAWC_i$$

Value

A numeric value of PAWC in mm

Examples

```
wheat <- read_apsimx(system.file("extdata/wheat.apsimx", package = "rapsimng"))

# Get total PAWC using LL15 (generic)
pawc <- get_pawc(wheat)
pawc

# Get PAWC for wheat crop (uses crop-specific LL and XF)
pawc_wheat <- get_pawc(wheat, crop = "Wheat")
pawc_wheat

# Get PAWC to 1200 mm depth
pawc_1200 <- get_pawc(wheat, depth = 1200)
pawc_1200

# Get wheat PAWC to 1500 mm depth
pawc_wheat_1500 <- get_pawc(wheat, crop = "Wheat", depth = 1500)
pawc_wheat_1500
```

---

get_simulations *Get simulations for a factorial experiment*

Description

Get simulations for a factorial experiment

Usage

```
get_simulations(l)
```

Arguments

l	A list from read_apsim with Factorial.Permutation as root.

Value

A list with Factor as name and Levels as values

Examples

```
wheat <- read_apsimx(system.file("extdata/wheat.apsimx", package = "rapsimng"))
permutation <- search_path(wheat, path = "[Factors].Permutation")
get_simulations(permutation$node)
```

`insert_model` *Insert a model into apsimx*

Description

Insert a model into apsimx

Usage

```
insert_model(l, path, model)
```

Arguments

`l`	the list of apsimx file
`path`	If numeric, the path returned by search_path or search_node. If character, the path supported by apsimx
`model`	A new model

Value

The modified list with new value

Examples

```
wheat <- read_apsimx(system.file("extdata/wheat.apsimx", package = "rapsimng"))
replacements <- new_model("Core.Replacements")
wheat_new <- insert_model(wheat, 1, replacements)
replacements_node <- search_path(wheat_new, ".Simulations.Replacements")
replacements_node$path
# Add a cultivar folder under replacements
cultivar_folder <- new_model("PMF.CultivarFolder", "Cultivars")
wheat_new <- insert_model(wheat_new, replacements_node$path, cultivar_folder)
cultivar_folder_node <- search_path(wheat_new,
                                    ".Simulations.Replacements.Cultivars")
cultivar_folder_node$path
# Add an new cultivar
cultivar <- new_model("PMF.Cultivar", "Hartog")
wheat_new <- insert_model(wheat_new, cultivar_folder_node$path, cultivar)
cultivar_node <- search_path(wheat_new,
                             ".Simulations.Replacements.Cultivars.Hartog")
cultivar_node$path
```

`insert_models` *Insert models into apsimx*

Description

Insert models into apsimx

Usage

```
insert_models(l, path, models)
```

Arguments

`l`	the list of apsimx file
`path`	If numeric, the path returned by search_path or search_node. If character, the path supported by apsimx
`models`	New models

Value

The modified list with new value

Examples

```
wheat <- read_apsimx(system.file("extdata/wheat.apsimx", package = "rapsimng"))
replacements <- new_model("Core.Replacements")
wheat_new <- insert_model(wheat, 1, replacements)
replacements_node <- search_path(wheat_new, ".Simulations.Replacements")
replacements_node$path
# Add a cultivar folder under replacements
cultivar_folder <- new_model("PMF.CultivarFolder", "Cultivars")
wheat_new <- insert_model(wheat_new, replacements_node$path, cultivar_folder)
cultivar_folder_node <- search_path(wheat_new,
                                    ".Simulations.Replacements.Cultivars")
cultivar_folder_node$path
# Add an new cultivar
cultivar <- new_model("PMF.Cultivar", "Hartog")
wheat_new <- insert_model(wheat_new, cultivar_folder_node$path, cultivar)
cultivar_node <- search_path(wheat_new,
                             ".Simulations.Replacements.Cultivars.Hartog")
cultivar_node$path
```

`install_apsimng` *Install ApsimNG Software*

Description

This function installs the ApsimNG simulation software on the local system. It handles the download and installation process for the APSIM Next Generation agricultural modeling platform.

Usage

```
install_apsimng(
  repo = "https://github.com/APSIMInitiative/ApsimX.git",
  branch = "master",
  install_dir = "apsimx_build",
  overwrite = FALSE
)
```

Arguments

`repo`	Character string specifying the repository of ApsimNG to install.
`branch`	Character string specifying the branch of the repository to install.
`install_dir`	Character string specifying the directory where ApsimNG should be installed.
`overwrite`	Logical. If TRUE, overwrites any existing installation at the specified install directory.

Details

This function downloads and installs ApsimNG from the official repository. It performs version checking, handles dependencies, and configures the installation for use with R. The function requires administrative privileges on Windows systems.

Value

No return value.

Examples

```
## Not run:
# Install latest version
install_apsimng()
install_apsimng(install_dir = "C:/MyPrograms/ApsimNG")

## End(Not run)
```

keep_simulations *Keep simulations for a factorial experiment*

Description

Keep simulations for a factorial experiment

Usage

```
keep_simulations(l, s)
```

Arguments

l	A list from read_apsim with Factorial.Permutation as root.
s	a list with factor as name and levels as value to keep. The factor is kept if it is not specified.

Value

A new list with removed simulations.

Examples

```
wheat <- read_apsimx(system.file("extdata/wheat.apsimx", package = "rapsimng"))
permutation <- search_path(wheat, path = "[Factors].Permutation")
permutation_new <- keep_simulations(permutation$node, list(V = "2"))
get_simulations(permutation_new)

permutation_new <- keep_simulations(permutation$node, list(Cv = c("Axe", "Bolac")))
get_simulations(permutation_new)

permutation_new <- keep_simulations(permutation$node,
                                    list(V = "1", Cv = c("Axe", "Bolac")))
get_simulations(permutation_new)
```

list_report *List all reports in the database*

Description

List all reports in the database

Usage

```
list_report(file)
```

Arguments

file	the file path to apsimx or db file

Value

a vector of all reports

Examples

```
## Not run:
file <- system.file("extdata/wheat.apsimx", package = "rapsimng")
list_report(file)

## End(Not run)
```

---

log_level *Set the log level of apsimx file*

Description

Set the log level of apsimx file

Usage

```
log_level(l, level = c("Error", "Warning", "Information", "Diagnostic", "All"))
```

Arguments

l	the list of apsimx file
level	log level with option Error, Warning, Information, Diagnostic, All

Value

a new apsimx file

minimum_apsimng *Create the minimum requirements to run an APSIM Next Generation*

Description

Create the minimum requirements to run an APSIM Next Generation

Usage

```
minimum_apsimng(install_path, output)
```

Arguments

install_path	The installed path of APSIM Next Generation
output	The output folder

Examples

```
## Not run:
 minimum_apsimng("C:/ProgramFiles/APSIMNG", "minimum_apsimng")

## End(Not run)
```

new_cultivar *Generate new cultivars with parameter which can be used in Replacements*

Description

Generate new cultivars with parameter which can be used in Replacements

Usage

```
new_cultivar(df, cultivar_folder = "Cultivars")
```

Arguments

df	A data frame for new parameters with three columns, i.e. name, parameter and value.
cultivar_folder	folder name for cultivars

Value

An APSIMX list

Examples

```
df <- data.frame(name = c("C1", "C1", "C2", "C2"),
     parameter = c("[Phenology].CAMP.FLNparams.MinLN",
               "[Phenology].CAMP.FLNparams.VrnLN",
               "[Phenology].CAMP.FLNparams.MinLN",
               "[Phenology].CAMP.FLNparams.VrnLN"),
     value = c(5, 6, 7, 8))
a <- new_cultivar(df)
```

---

new_model *Create a new model*

Description

Create a new model

Usage

```
new_model(model, name = model)
```

Arguments

model	The name of new model
name	The new name

Examples

```
new_model(model = "PMF.Cultivar")
new_model(model = "PMF.Cultivar", name = "example")
```

---

read_apsimx *Read APSIMX file*

Description

Read APSIMX file

Usage

```
read_apsimx(path)
```

Arguments

path	The file path or URL to apsimx file

Value

A list object of apsimx file

Examples

```
file <- system.file("extdata/wheat.apsimx", package = "rapsimng")
m <- read_apsimx(file)
```

`read_report` *Read apsimx database in db file format*

Description

Read apsimx database in db file format

Usage

```
read_report(file, report)
```

Arguments

`file`	the file path to apsimx or db file
`report`	the report name

Value

a data.frame for a report

Examples

```
## Not run:
file <- system.file("extdata/wheat.apsimx", package = "rapsimng")
read_report(file, "HarvestReport")

## End(Not run)
```

remove_model *Remove a model with new values*

Description

Remove a model with new values

Usage

```
remove_model(l, path)
```

Arguments

l	the list of apsimx file
path	If numeric, the path returned by search_path or search_node. If character, the path supported by apsimx

Value

The modified list with new value

Examples

```
wheat <- read_apsimx(system.file("extdata/Wheat.json", package = "rapsimng"))
a <- search_path(wheat, '[Wheat].Phenology.ThermalTime')
wheat_new <- remove_model(wheat, a$path)
b <- search_path(wheat_new, '[Wheat].Phenology.ThermalTime')
b
```

replace_model *Replace a model with new values*

Description

Replace a model with new values

Usage

```
replace_model(l, path, model)
```

Arguments

l	the list of apsimx file
path	If numeric, the path returned by search_path or search_node. If character, the path supported by apsimx
model	A new model

Value

The modified list with new value

Examples

```
wheat <- read_apsimx(system.file("extdata/Wheat.json", package = "rapsimng"))

a <- search_path(wheat, '[Wheat].Phenology.ThermalTime')
a$node$Children[[1]]$X[[2]] <- 27
wheat_new <- replace_model(wheat, a$path, a$node)
b <- search_path(wheat_new, '[Wheat].Phenology.ThermalTime')
b$node$Children[[1]]$X
```

---

run_models *Run apsimx file using Models.exe*

Description

Run apsimx file using Models.exe

Usage

```
run_models(
  models_exe,
  path,
  pattern = NULL,
  recurse = FALSE,
  csv = FALSE,
  parallel = NULL,
  ncpus = NULL,
  verbose = FALSE
)
```

Arguments

models_exe	path to Models.exe
path	The path to an .apsimx file. May include wildcard.
pattern	Use to filter simulation names to run.
recurse	Recursively search subdirectories for files matching ApsimXFileSpec. FALSE in default.
csv	Export all reports to .csv files. FALSE in default.
parallel	Use the multi-process job runner. If FALSE, use single threaded; if TRUE, use the multi-process job runner
ncpus	Set the number of processors to use. All processes in default
verbose	Write messages to StdOut when a simulation starts/finishes. Only has an effect when running a directory of .apsimx files (*.apsimx).

search_node *Find element(s) in apsimx file*

Description

Find element(s) in apsimx file

Usage

```
search_node(l, all = FALSE, max_depth = 1e+06, case_insensitive = TRUE, ...)
```

Arguments

l	The list of apsimx file
all	Whether to find all elements
max_depth	The maximum depth to search
case_insensitive	Whether case sensitive
...	Other names arguments for property to match

Value

A list matching all criteria if all equals to TRUE, A list with node and path if all equals to FALSE (default)

Examples

```
wheat <- read_apsimx(system.file("extdata/Wheat.json", package = "rapsimng"))
# Return empty list if not found
search_node(wheat, Name = "Simulations1")
# Find root level
a <- search_node(wheat, Name = "Simulations")
a$path
# Find sub-level
a <- search_node(wheat, Name = "Wheat")
a$path
a <- search_node(wheat, `$type` = "Models.PMF.Cultivar, Models")
a$path

# Find multiple attributes
a <- search_node(wheat,
            Name = 'PotentialBranchingRate',
            `$type` = "Models.Functions.PhaseLookup, Models")
a$path
a$node$Name
# Find all cultivar nodes
 a <- search_node(wheat, `$type` = "Models.PMF.Cultivar, Models", all = TRUE)
length(a)
```

search_path *Find a model in the apsimx file using specified path*

Description

Find a model in the apsimx file using specified path

Usage

```
search_path(l, path, case_insensitive = TRUE)
```

Arguments

l	the list of apsimx file
path	The specified path (See details)
case_insensitive	Whether case sensitive

Value

The list for the specified path.

Absolute Paths

Absolute paths have a leading '.' e.g.

- .Simulations.Test.Clock - absolute path - refers to the clock model in the 'Test' simulation.

Scoped Paths

Scoped paths have a leading model type in square brackets. A model of the specified name, in scope, is located before applying the rest of the path.

- [Soil].Water - scoped path - refers to the Water model that is a child of a model that has the name 'Soil' that is in scope

Examples

```
wheat <- read_apsimx(system.file("extdata/Wheat.json", package = "rapsimng"))
# Return empty list if not found
search_path(wheat, "[Simulations1]")
# Search root path
a <- search_path(wheat, '.Simulations')
a$path
a$node$Name
# Level one
a <- search_path(wheat, '.Simulations.Wheat1')
a$path
a$node$Name
```

```
# Level two
a <- search_path(wheat, '.Simulations.Wheat')
a$path
a$node$Name
# Level three
a <- search_path(wheat, '.Simulations.Wheat.BranchingRate')
a$path
a$node$Name
a <- search_path(wheat, '.Simulations.Wheat.Structure')
a$path
a$node$Name
# Level four
a <- search_path(wheat, '.Simulations.Wheat.Structure.BranchingRate')
a$path
a$node$Name
a <- search_path(wheat, '.Simulations.Wheat.Structure.BranchingRate1')
a$path
a$node$Name
# scoped
# Root path
a <- search_path(wheat, '[Simulations1]')
a <- search_path(wheat, '[Simulations]')
a$path
a$node$Name
# Level two
a <- search_path(wheat, '[Simulations].Wheat1')
a <- search_path(wheat, '[Simulations1].Wheat')
a$path
a$node$Name
a <- search_path(wheat, '[Whea]')
a <- search_path(wheat, '[Wheat]')
a$path
a$node$Name
# Level three
a <- search_path(wheat, '[Wheat].BranchingRate')
a <- search_path(wheat, '[Wheat].Structure')
a$path
a$node$Name
a <- search_path(wheat, '[Structure]')
a$path
a$node$Name
# Level four
a <- search_path(wheat, '[Structure].BranchingRate')
a$path
a$node$Name
a <- search_path(wheat, '[Structure].BranchingRate1')
a <- search_path(wheat, '[Structure1].BranchingRate')
```

---

set_parameter_value *Set a parameter with a new value*

Description

Set a parameter with a new value

Usage

```
set_parameter_value(l, parameter, value)
```

Arguments

l	the list of apsimx file
parameter	the name of parameter with APSIM NG specification
value	the new value

Value

A list with replaced value

Examples

```
 wheat <- read_apsimx(system.file("extdata/Wheat.json", package = "rapsimng"))
 new_wheat <- set_parameter_value(wheat,
  "[Structure].BranchingRate.PotentialBranchingRate.Reproductive.Zero.FixedValue",
  1)
new_wheat2 <- search_path(new_wheat,
    "[Structure].BranchingRate.PotentialBranchingRate.Reproductive.Zero")
new_wheat2$node$FixedValue

new_wheat <- set_parameter_value(
    wheat,
    "[Structure].HeightModel.WaterStress.XYPairs.Y",
    "0.1,1.1")
new_wheat2 <- search_path(new_wheat,
    "[Structure].HeightModel.WaterStress.XYPairs")
new_wheat2$node$Y
```

---

split_apsimx *Split an APSIMX file into separate simulations except cultivar*

Description

This function takes an APSIMX file and splits it into separate simulations for all factors except cultivar factors. Then saving the results to the specified output location.

Usage

```
split_apsimx(file, output)
```

Arguments

file	Character. Path to the input APSIMX file.
output	Character. Path to the directory where the split components will be saved.

Value

No return value. The function is called for its side effects.

Examples

```
## Not run:
split_apsimx("path/to/input.apsimx", "path/to/output/directory")

## End(Not run)
```

---

test_apsimx *Test whether all files under published folder of apsimx are required*

Description

Test whether all files under published folder of apsimx are required

Usage

```
test_apsimx(base, example)
```

Arguments

base	the base folder path to apsimx publish
example	an example apsimx file

Value

A vector a required files

update_cultivar *Title Update the cultivar parameters*

Description

This function assumes the file is apsimx format. A new Replacements node is added if it is not exist. The existing cultivar parameters are updated. New cultivar is created.

Usage

```
update_cultivar(
  l,
  df,
  add = TRUE,
  use_folder = TRUE,
  cultivar_folder = "Cultivars"
)
```

Arguments

l	The list of apsimx file
df	A data frame for new parameters with three columns, i.e. name, parameter and value.
add	Whether to add extra nodes (e.g. replacements, Cultivars folder and new cultivar)
use_folder	use cultivar folder to add new cultivars
cultivar_folder	folder name for cultivars

Value

The modified apsimx file

Examples

```
wheat <- read_apsimx(system.file("extdata/wheat.apsimx", package = "rapsimng"))
# Update cultivars
df <- data.frame(name = rep("Hartog", 3),
                 parameter = c("[Phenology].MinimumLeafNumber.FixedValue",
                               "[Phenology].VrnSensitivity.FixedValue",
                               "[Phenology].PpSensitivity.FixedValue"),
                value = c(9, 7, 3))

wheat_cultivar <- update_cultivar(wheat, df)
hartog <- search_path(wheat_cultivar, "[Replacements].Hartog")
hartog$path
```

`with_apsimx` *Run APSIM Next Generation Simulations in a Temporary Folder*

Description

The `with_apsimx` function automates the process of setting up and running APSIM Next Generation (NG) simulations in a temporary or specified folder. This approach helps reduce I/O overhead, particularly in high-performance computing (HPC) environments with network file systems (NFS). The frequent file I/O operations performed by APSIM NG (e.g., writing to SQLite databases) can strain file systems, and redirecting simulations to a local folder mitigates this issue. The function can also perform optional post-processing of the simulation results.

Usage

```
with_apsimx(
  models,
  file,
  mets = NULL,
  target = tempdir(),
  clean = c("none", "simulations", "mets", "all"),
  post_process = NULL,
  ...
)
```

Arguments

`models`	A character string specifying the path to the APSIM NG executable (`Models.exe` on Windows or `Models` on Linux).
`file`	A character string specifying the path to the `.apsimx` simulation file.
`mets`	A character vector specifying paths to meteorological data (`.met` files) used in the simulation. Currently, only `.met` files located in the same folder as the `.apsimx` file are supported (optional).
`target`	A character string specifying the target directory where simulations will be run. Defaults to the R system temporary directory (`tempdir()`).
`clean`	A character string specifying which files or directories to clean after the simulation. Options are: • `"none"`: No files are cleaned (default). • `"simulations"`: Cleans only simulation-related files (e.g., `*.apsimx`, `*.db`, `*.db-shm`, `*.db-wal`, `*.csv`). • `"mets"`: Cleans only meteorological files (e.g., `*.met`). • `"all"`: Cleans all files, including APSIM NG executable files, simulations, and meteorological files.
`post_process`	An optional function for post-processing simulation results. The function must accept a `folder` argument specifying the directory containing the simulation results.

... Additional arguments passed to both the run_models and post_process functions.

Value

If a post_process function is provided, its return value is returned. Otherwise, the function returns NULL.

See Also

run_models: Runs APSIM simulations. do.call: Dynamically calls functions with named arguments.

Examples

```
## Not run:
# Run simulations without post-processing
with_apsimx(
    models = "path/to/apsimx",
    file = "path/to/input.apsimx",
    mets = c("path/to/met1.met", "path/to/met2.met"),
    clean = "all"
)

# Run simulations with post-processing
post_process_function <- function(folder) {
    output_files <- list.files(folder, full.names = TRUE)
    message("Output files:", paste(output_files, collapse = "\n"))
}

result <- with_apsimx(
    models = "path/to/apsimx",
    file = "path/to/input.apsimx",
    mets = c("path/to/met1.met", "path/to/met2.met"),
    post_process = post_process_function
)

## End(Not run)
```

---

write_apsimx *Write APSIMX file*

Description

Write APSIMX file

Usage

```
write_apsimx(l, file)
```

Arguments

`l`	the list of apsimx file
`file`	The file path to apsimx file

Value

A list object of apsimx file

Index