

# Package ‘apsimx’

July 10, 2020

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

**Version** 1.946

**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.com/>>).

**Depends** R (>= 3.3.0)

**License** GPL-3

**Encoding** UTF-8

**VignetteBuilder** knitr

**BugReports** <https://github.com/femiguez/apsimx/issues>

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

**Suggests** datasets, FedData, ggplot2, listviewer, nasapower, raster, reactR, rmarkdown, sp, spData, sf

**LazyData** true

**RoxygenNote** 7.1.0

**NeedsCompilation** no

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

**Maintainer** Fernando Miguez <[femiguez@iastate.edu](mailto:femiguez@iastate.edu)>

**Repository** CRAN

**Date/Publication** 2020-07-09 22:00:03 UTC

## R topics documented:

apsim . . . . .	3
apsim.options . . . . .	4

apsimx . . . . .	4
apsimx.options . . . . .	5
apsimx_example . . . . .	6
apsimx_filetype . . . . .	7
apsimx_options . . . . .	8
apsimx_soil_profile . . . . .	9
apsim_example . . . . .	11
apsim_options . . . . .	12
apsim_version . . . . .	13
auto_detect_apsimx_examples . . . . .	14
auto_detect_apsim_examples . . . . .	14
check_apsim_met . . . . .	15
edit_apsim . . . . .	16
edit_apsimx . . . . .	17
edit_apsimx_batch . . . . .	20
edit_apsimx_replacement . . . . .	21
edit_apsimx_replace_soil_profile . . . . .	23
edit_apsim_replace_soil_profile . . . . .	24
edit_apsim_xml . . . . .	26
get_daymet_apsim_met . . . . .	27
get_iemre_apsim_met . . . . .	29
get_iem_apsim_met . . . . .	30
get_power_apsim_met . . . . .	31
impute_apsim_met . . . . .	32
inspect_apsim . . . . .	32
inspect_apsimx . . . . .	34
inspect_apsimx_replacement . . . . .	36
print.met . . . . .	38
read_apsim . . . . .	38
read_apsimx . . . . .	39
read_apsimx_all . . . . .	39
read_apsim_all . . . . .	40
read_apsim_met . . . . .	41
soilwat_parms . . . . .	42
ssurgo2sp . . . . .	43
swim_parms . . . . .	44
unit_conv . . . . .	46
view_apsim . . . . .	47
view_apsimx . . . . .	48
write_apsim_met . . . . .	48

---

`apsim`*Run an APSIM (7.x) 'Classic' simulation*

---

## Description

Run `apsim` from R. It's for Windows only. It uses 'shell'.

## Usage

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

## Arguments

<code>file</code>	file name to be run (the extension <code>.apsim</code> is optional)
<code>src.dir</code>	directory containing the <code>.apsim</code> file to be run (defaults to the current directory)
<code>silent</code>	whether to print messages from <code>apsim</code> simulation
<code>value</code>	how much output to return: option 'all' returns all components of the simulation; option 'report' returns only the 'main' report component; option 'none' does not create a data.frame but it generates the databases from APSIM-X
<code>cleanup</code>	logical. Whether to delete the <code>.out</code> file generated by APSIM. Default is FALSE.

## 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

## Examples

```
## Not run:  
## See function 'apsim_example'  
  
## End(Not run)
```

---

apsim.options	<i>Environment which stores APSIM options</i>
---------------	---

---

### 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 undesirable consequences.

### Usage

```
apsim.options
```

### Format

An object of class environment of length 3.

### Details

Environment which stores APSIM options

### Examples

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

## End(Not run)
```

---

apsimx	<i>Run an APSIM-X simulation</i>
--------	----------------------------------

---

### Description

Run apsimx from R. It uses 'system' (unix) or 'shell' (windows) and it attempts to be platform independent.

### Usage

```
apsimx(
  file = "",
  src.dir = ".",
  silent = FALSE,
  value = c("all", "report", "none"),
  cleanup = FALSE
)
```

**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)
silent	whether to print messages from apsim simulation
value	how much output to return: option 'all' returns all components of the simulation; option 'report' returns only the 'main' report component; option 'none' does not create a data.frame but it generates the databases from APSIM-X
cleanup	logical. Whether to delete the .db file generated by APSIM-X. Default is FALSE

**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. To change the variables being reported use the GUI.

**Examples**

```
## Not run:
## See function 'apsimx_example' and vignette 'apsimx'

## End(Not run)
```

---

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

**Examples**

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

## End(Not run)
```

---

apsimx_example	<i>Access Example APSIM-X Simulations</i>
----------------	---

---

**Description**

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

**Usage**

```
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.

**Note**

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

## Examples

```
## 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	<i>Test file format for .apsimx files</i>
-----------------	---

---

## Description

Test whether an .apsimx file is XML or json

## Usage

```
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

```
## Not run:
ex.dir <- auto_detect_apsimx_examples()
apsimx_filetype("Barley", src.dir = ex.dir)
apsimx_filetype("Chicory", src.dir = ex.dir)
apsimx_filetype("Oats", src.dir = ex.dir)
apsimx_filetype("Maize", src.dir = ex.dir)
apsimx_filetype("Sugarcane", src.dir = ex.dir)
```

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

---

```
apsimx_options      Setting some options for the package
```

---

## Description

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

## Usage

```
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

## 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

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

## End(Not run)
```



---

 apsimx\_soil\_profile *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

```
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)
)

## S3 method for class 'soil_profile'
plot(
```

```

x,
...,
property = c("all", "water", "BD", "AirDry", "LL15", "DUL", "SAT", "KS", "crop.XF",
"crop.KL", "crop.LL", "Carbon", "SoilCNRatio", "FOM", "FOM.CN", "FBiom", "FInert",
"NO3N", "NH4N", "PH")
)

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 <a href="#">inspect_apsimx</a> with soil.child = “Metadata”.
soilwat	optional ‘list’ of class ‘soilwat_parms’
swim	optional ‘list’ of class ‘swim_parms’

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 $a * \text{soil.layer} * \exp(-b * \text{soil.layer})$ .
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' and 'crops' (for now)

## Examples

```
## Not run:
sp <- apsimx_soil_profile()
require(ggplot2)
plot(sp)

## End(Not run)
```

## Description

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

**Usage**

```
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.

**Note**

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

**Examples**

```
## 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**

```
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

**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**

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

## End(Not run)
```

---

apsim_version	<i>Display available APSIM ‘Classic’ and APSIM-X versions</i>
---------------	---

---

**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**

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

---

auto\_detect\_apsimx\_examples

*Auto detect where apsimx examples are located*

---

**Description**

simple function to detect where APSIM-X examples are located

**Usage**

```
auto_detect_apsimx_examples()
```

**Details**

Auto detect where apsimx examples are located

**Value**

will create a directory pointing to APSIM-X distributed examples

**Examples**

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

---

auto\_detect\_apsim\_examples

*Auto detect where apsim examples are located*

---

**Description**

simple function to detect where APSIM 'Classic' examples are located

**Usage**

```
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**

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

---

check_apsim_met	<i>Check a met file for possible errors</i>
-----------------	---

---

**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/m<sup>2</sup>/day);  
'rain': range (0 to 100) – units (mm/day)

**Value**

does not return anything unless possible errors are found

---

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,
  node = c("Clock", "Weather", "Soil", "SurfaceOrganicMatter", "MicroClimate", "Crop",
    "Manager", "Other"),
  soil.child = c("Metadata", "Water", "OrganicMatter", "Chemical", "Analysis",
    "InitialWater", "Sample", "SWIM"),
  manager.child = NULL,
  parm = NULL,
  value = NULL,
  overwrite = FALSE,
  edit.tag = "-edited",
  parm.path = NULL,
  verbose = 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
node	either 'Clock', 'Weather', 'Soil', 'SurfaceOrganicMatter', 'MicroClimate', 'Crop', 'Manager' or 'Other'
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'
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. The function is similar to the `edit_apsim` function in the `'apsimr'` package, but with the difference that only some variables (parameters) can be modified.

The function `inspect_apsim` is for a quick look from within R. The APSIM GUI provides a more complete examination of the `.apsim` 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 (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

```
## 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")
```

---

edit\_apsimx

*Edit an APSIM-X (JSON) Simulation*

---

## Description

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

**Usage**

```

edit_apsimx(
  file,
  src.dir = ".",
  wrt.dir = NULL,
  node = c("Clock", "Weather", "Soil", "SurfaceOrganicMatter", "MicroClimate", "Crop",
           "Manager"),
  soil.child = c("Metadata", "Water", "Organic", "Physical", "Analysis", "Chemical",
                 "InitialWater", "Sample"),
  manager.child = NULL,
  parm = NULL,
  value = NULL,
  overwrite = FALSE,
  edit.tag = "-edited",
  parm.path = NULL,
  root,
  verbose = TRUE
)

```

**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' 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. The function is similar to the edit\_apsim

functin in the ‘apsimr’ package, but with the difference that only some variables (parameters) can be modified.

The function inspect\_apsimx is for a quick look from within R. The APSIM GUI provides a more complete examination of the .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 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

```
## Not run:
## This example will read one of the examples distributed with APSIM-X
## but write to a temporary directory
tmp.dir <- tempdir()

## Edit Bulk density
ex.dir <- auto_detect_apsimx_examples()
bds <- c(1.02, 1.03, 1.09, 1.16, 1.18, 1.19, 1.20)
edit_apsimx("Barley.apsimx", src.dir = ex.dir,
            wrt.dir = tmp.dir,
            node = "Soil",
            soil.child = "Water",
            parm = "BD", value = bds,
            verbose = FALSE)

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

## To delete the file...
file.remove(paste0(tmp.dir, "/Barley-edited.apsimx"))

## Edit the fertilizer amount in 'Maize.apsimx'
edit_apsimx("Maize.apsimx", src.dir = ex.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(paste0(tmp.dir, "/Maize-edited.apsimx"))

## End(Not run)
```

---

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

```
edit_apsimx_batch(
    file,
    src.dir = ".",
    wrt.dir = NULL,
    parms = NULL,
    silent = FALSE,
    verbose = TRUE
)
```

## 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 for of '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.

```
[Clock].StartDate = 2019-1-20 .Simulations.Sim1.Name = SimulationVariant35 .Simulations.Sim2.Enabled = false .Simulations.Sim1.Paddock.Soil.Thickness[1] = 50 Notes:
```

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. ie [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

```
## Not run:
## This example will read one of the examples distributed with APSIM-X
## but write to a temporary directory

tmp.dir <- tempdir()

## Edit InitialResidueMass
extd.dir <- system.file("extdata", package = "apsimx")
parms <- list(`.Simulations.Simulation.Field.SurfaceOrganicMatter.InitialResidueMass` = 600)
edit_apsimx_batch("Wheat.apsimx", src.dir = extd.dir, wrt.dir = tmp.dir, parms = parms)

## End(Not run)
```

---

```
edit_apsimx_replacement
```

*Edit a replacement component in an .apsimx (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

```
edit_apsimx_replacement(
  file = "",
  src.dir = ".",
  wrt.dir = ".",
  node = NULL,
  node.child = NULL,
  node.subchild = NULL,
  node.subsubchild = NULL,
  node.sub3child = NULL,
  node.string = NULL,
  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 be inspected.
node.string	passing of a string instead of the node hierarchy.
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 because 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()

edit_apsimx_replacement("MaizeSoybean.apsimx",
                        src.dir = extd.dir, wrt.dir = tmp.dir,
                        node = "Maize", node.child = "Phenology",
                        node.subchild = "ThermalTime", parm = c("X", "Y"),
                        value = c(1,2,3,4,5))

```

---

```

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 = "",
  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**

```
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")
```

---

edit\_apsim\_replace\_soil\_profile

*Edit APSIM 'Classic' file with a replaced soil profile*

---

**Description**

Edits an APSIM Classic simulation by replacing the soil profile

**Usage**

```
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 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. 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**

```
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	<i>Edit an APSIM (Classic) Simulation auxiliary xml file</i>
----------------	--

---

### Description

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

### Usage

```
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 .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
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 .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. The function is similar to the edit\_apsim function in the 'apsimr' package, but with the difference that only some variables (parameters) can be modified.

The function inspect\_apsim is for a quick look from within R. The APSIM GUI provides a more complete examination of the .apsim file

**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 no inspect equivalent, so you need to inspect the file manually and/or use some trial and error to find the correct parameter and double check that it was edited correctly. It is more flexible than 'edit\_apsim' and (perhaps) equivalent to 'apsimr::edit\_sim\_file'.

**Examples**

```
## 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)
```

---

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**

```
get_daymet_apsim_met(
  lonlat,
  years,
  wrt.dir = ".",
  filename = NULL,
  width.height = c(0.001, 0.001),
  template,
  label = NULL,
  elements = NULL,
```



```
summary(dmet12)
## Check for reasonable ranges
check_apsim_met(dmet12)

## 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)
```

## Arguments

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

## 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.

## Examples

```
## 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
summary(iemre)

## 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)
```

### Arguments

lonlat	Longitude and latitude vector (optional)
dates	date ranges
wrt.dir	write directory
state	state which you choose climate data from
station	station which you choose climate data from
filename	file name for writing out to disk

### Details

The original data can be obtained from: <https://mesonet.agron.iastate.edu/request/coop/fe.phtml>

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. For this function either provide the longitude and latitude or the state and station, but not both. In fact, 'state' and 'station' will be ignored if 'lonlat' is supplied.

### Examples

```
## Not run:
## This will not write a file to disk
iem.met <- get_iem_apsim_met(state = "IA",
                           station = "IA5198",
                           dates = c("2012-01-01", "2012-12-31"))

summary(iem.met)

## Alternatively, coordinates can be used
## This should be equivalent to the previous request
iem.met2 <- get_iem_apsim_met(lonlat = c(-93,42),
                             dates = c("2012-01-01", "2012-12-31"))

summary(iem.met2)

## End(Not run)
```

---

get\_power\_apsim\_met     *Get 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.

## Examples

```
## 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"))
## Note that missing data is coded as -99
summary(pwr)
## Check for reasonable ranges
check_apsim_met(pwr)
## replace -99 with NA
pwr$radn <- ifelse(pwr$radn == -99, NA, pwr$radn)
## Impute using linear interpolation
pwr.imptd <- impute_apsim_met(pwr, verbose = TRUE)
summary(pwr.imptd)

## End(Not run)
```

---

impute_apsim_met	<i>Perform imputation for missing data in a met file</i>
------------------	--

---

**Description**

Takes in an object of class 'met' and imputes values

**Usage**

```
impute_apsim_met(met, method = c("approx", "splines", "mean"), verbose = FALSE)
```

**Arguments**

met	object of class 'met'
method	method for imputation, <a href="#">approx</a> , <a href="#">splines</a> or <a href="#">mean</a>
verbose	whether to print missing data to the console, default = FALSE

**Value**

an object of class 'met' with attributes

---

inspect_apsim	<i>Inspect an .apsim (XML) file</i>
---------------	-------------------------------------

---

**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 = ".",
  node = c("Clock", "Weather", "Soil", "SurfaceOrganicMatter", "Crop", "Manager",
    "Other"),
  soil.child = c("Metadata", "Water", "OrganicMatter", "Nitrogen", "Analysis",
    "InitialWater", "Sample", "SWIM"),
  parm = NULL,
  digits = 3,
  print.path = FALSE
)
```



**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
node	either 'Weather', 'Soil', 'SurfaceOrganicMatter', 'MicroClimate', 'Crop', 'Manager' or 'Other'
soil.child	specific soil component to be inspected
parm	parameter to inspect when node = 'Crop', 'Manager' or 'Other'
digits	number of decimals to print (default 3)
print.path	whether to print the parameter path (default = FALSE)

**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**

table with inspected parameters and values

**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 = "Water")
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 = "SurfaceOrganicMatter")
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))

## 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 = "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)

```

---

inspect\_apsimx

*Inspect an .apsimx (JSON) file*


---

## 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 = ".",
  node = c("Clock", "Weather", "Soil", "SurfaceOrganicMatter", "MicroClimate", "Crop",
          "Manager", "Other"),
  soil.child = c("Metadata", "Water", "InitialWater", "Chemical", "Physical",
                "Analysis", "SoilWater", "InitialN", "CERESSoilTemperature", "Sample", "Nutrient",

```

```

    "Organic"),
  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	specific node to be inspected either 'Clock', 'Weather', 'Soil', 'SurfaceOrganicMatter', 'MicroClimate', 'Crop', 'Manager' or 'Other'
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

```

## Not run:
ex.dir <- auto_detect_apsimx_examples()
inspect_apsimx("Barley", src.dir = ex.dir, node = "Clock")
inspect_apsimx("Barley", src.dir = ex.dir, node = "Weather")
inspect_apsimx("Barley", src.dir = ex.dir, node = "Soil", soil.child = "Metadata")
inspect_apsimx("Barley", src.dir = ex.dir, node = "Soil", soil.child = "Water")
inspect_apsimx("Barley", src.dir = ex.dir, node = "Soil", soil.child = "SoilWater")
inspect_apsimx("Barley", src.dir = ex.dir, node = "Soil", soil.child = "Organic")
inspect_apsimx("Barley", src.dir = ex.dir, node = "Soil", soil.child = "Chemical")
inspect_apsimx("Barley", src.dir = ex.dir, node = "Soil", soil.child = "InitialWater")

```

```
inspect_apsimx("Barley", src.dir = ex.dir, node = "Soil", soil.child = "InitialN")
inspect_apsimx("Barley", src.dir = ex.dir, node = "SurfaceOrganicMatter")
inspect_apsimx("Barley", src.dir = ex.dir, node = "MicroClimate")
inspect_apsimx("Barley", src.dir = ex.dir, node = "Crop")
inspect_apsimx("Barley", src.dir = ex.dir, node = "Manager")

## End(Not run)
```

---

```
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.

### Usage

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

### 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 sub-sub-subchild 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)
print.path	print the path to the inspected parameter (default FALSE)
verbose	whether to print additional information, default: TRUE

### 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

### 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 = "TemperatureResponse")

## Not run:
## This function can also be used to inspect more complex APSIM-X files
## For example the 'Factorial' example
ex.dir <- auto_detect_apsimx_examples()
inspect_apsimx_replacement("Factorial", src.dir = ex.dir,
                           root = list("Experiment", 1), node = "Base", node.child = "Weather")

## End(Not run)
```

---

print.met	<i>Printer-friendly version of a metfile</i>
-----------	--

---

**Description**

Print a met file in a friendly way

**Usage**

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

**Arguments**

x	an R object of class 'met'
...	additional printing arguments

---

read_apsim	<i>Read APSIM generated .out files</i>
------------	--

---

**Description**

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

**Usage**

```
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

**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	<i>Read APSIM-X generated .db files</i>
-------------	---

---

**Description**

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

**Usage**

```
read_apsimx(file = "", src.dir = ".", value = c("report", "all"))
```

**Arguments**

file	file name
src.dir	source directory where file is located
value	either 'report' (data.frame) or 'all' (list)

**Details**

Read APSIM-X generated .db files

---

read_apsimx_all	<i>Read all APSIM-X generated .db files in a directory</i>
-----------------	--

---

**Description**

Like [read\\_apsimx](#), but it reads all .db files in a directory.

**Usage**

```
read_apsimx_all(src.dir = ".", value = c("report", "all"))
```

**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

**Note**

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

---

read_apsim_all	<i>Read all APSIM generated .out files in a directory</i>
----------------	---

---

**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"  
)
```

**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

**Details**

Read all APSIM generated .out files in a directory

**Note**

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



---

read_apsim_met	<i>Read in an APSIM met file</i>
----------------	----------------------------------

---

## 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

```
## Not run:  
extd.dir <- system.file("extdata", package = "apsimx")  
ames.met <- read_apsim_met("Ames.met", src.dir = extd.dir)  
ames.met  
  
## End(Not run)
```

---

`soilwat_parms`*Helper function to supply SoilWat parameters*

---

### Description

Creates a list with specific components for the SoilWat model

### 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  
)
```

### 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

### Details

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

**Value**

a 'list' with class 'soilwat\_parms'

---

ssurgo2sp

*Take in SSURGO csv files and create a soil profile*


---

**Description**

Utility function to convert SSURGO data to soil profile

**Usage**

```
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 <a href="#">approx</a> )
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**

```
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")
```

**Description**

Creates a list with specific components for the SWIM model

**Usage**

```
swim_parms(
  Salb = NA,
  CN2Bare = NA,
  CNRed = NA,
  CNCov = NA,
  KDul = NA,
  PSIDul = NA,
  VC = NA,
  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'

---

unit_conv	<i>performs common unit conversions</i>
-----------	---

---

### Description

Function which performs common unit conversions

### Usage

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

### Arguments

x	input variable
from	original units
to	target units

### Details

At the moment possible conversions are:

- 'g/m2' to 'kg/ha'
- 'kg/ha' to 'g/m2'

**Value**

value of the input variable with new units

**Examples**

```
## Not run:
grain.yield.gm2 <- 600
grain.yield.kgha <- unit_conv(grain.yield.gm2, from = "g/m2", to = "kg/ha")
grain.yield.kgha

## End(Not run)
```

---

view\_apsim

*Viewing an APSIM Classic file interactively*


---

**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 <b>listviewer</b> .

**Note**

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

**Examples**

```
## Not run:
extd.dir <- system.file("extdata", package = "apsimx")
## View the structure of the APSIM-X simulation file
view_apsim("Millet.apsim", src.dir = extd.dir)

## End(Not run)
```

---

view_apsimx	<i>Viewing an APSIM-X file interactively</i>
-------------	--

---

**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 <b>listviewer</b> .

**Note**

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

**Examples**

```
## Not run:
extd.dir <- system.file("extdata", package = "apsimx")
## View the structure of the APSIM-X simulation file
view_apsimx("Wheat.apsimx", src.dir = extd.dir)

## End(Not run)
```

---

write_apsim_met	<i>Write an APSIM met file</i>
-----------------	--------------------------------

---

**Description**

Write an object of class 'met' to disk

**Usage**

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



**Arguments**

<code>met</code>	object of class 'met'
<code>wrt.dir</code>	directory where the file will be written
<code>filename</code>	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**

```
## Not run:
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

## End(Not run)
```

# Index

- \* **datasets**
  - apsim.options, 4
  - apsimx.options, 5
- approx, 32, 43
- apsim, 3
- apsim.options, 4
- apsim\_example, 11
- apsim\_options, 12
- apsim\_version, 13
- apsimx, 4
- apsimx.options, 5
- apsimx\_example, 6
- apsimx\_filetype, 7
- apsimx\_options, 8
- apsimx\_soil\_profile, 9, 11
- auto\_detect\_apsim\_examples, 14
- auto\_detect\_apsimx\_examples, 14
  
- check\_apsim\_met, 15
- check\_apsimx\_soil\_profile  
    (apsimx\_soil\_profile), 9
  
- edit\_apsim, 16
- edit\_apsim\_replace\_soil\_profile, 24
- edit\_apsim\_xml, 26
- edit\_apsimx, 17
- edit\_apsimx\_batch, 20
- edit\_apsimx\_replace\_soil\_profile, 23
- edit\_apsimx\_replacement, 21
  
- get\_daymet, 27, 28
- get\_daymet\_apsim\_met, 27
- get\_iem\_apsim\_met, 30
- get\_iemre\_apsim\_met, 29
- get\_power, 31
- get\_power\_apsim\_met, 31
  
- impute\_apsim\_met, 32
- inspect\_apsim, 32
- inspect\_apsimx, 10, 34
  
- inspect\_apsimx\_replacement, 36
  
- mean, 32
  
- plot.soil\_profile  
    (apsimx\_soil\_profile), 9
- print.met, 38
  
- read\_apsim, 38, 40
- read\_apsim\_all, 40
- read\_apsim\_met, 41
- read\_apsimx, 39, 39
- read\_apsimx\_all, 39
  
- soilwat\_parms, 42
- splines, 32
- ssurgo2sp, 43
- swim\_parms, 44
  
- unit\_conv, 46
  
- view\_apsim, 47
- view\_apsimx, 48
  
- write\_apsim\_met, 48