

# Package ‘Maeswrap’

July 2, 2014

**Type** Package

**Title** Wrapper functions for MAESTRA/MAESPA.

**Version** 1.4.11

**Date** 2012-05-25

**Author** Remko Duursma

**Maintainer** Remko Duursma <remkoduursma@gmail.com>

**Depends** rgl,lattice,geometry

**Description** A bundle of functions for modifying MAESTRA/MAESPA input files, reading output files, and visualizing the stand in 3D. Handy for running sensitivity analyses, scenario analyses, etc.

**License** GPL

**LazyLoad** yes

**Repository** CRAN

**Date/Publication** 2012-05-25 06:19:42

**NeedsCompilation** no

## R topics documented:

Maeswrap-package . . . . .	2
addarrow . . . . .	3
BatchUtilities . . . . .	4
checkwatbal . . . . .	6
maeswrapdefinitions . . . . .	7
Plotstand . . . . .	7
plotuspar . . . . .	9
readdayflux . . . . .	11
readhrflux . . . . .	11

readmet . . . . .	12
readPAR . . . . .	13
readwatbal . . . . .	14
replacemetvar . . . . .	15
replaceNAMELIST . . . . .	16
replacePAR . . . . .	17
revchar . . . . .	18
runfiletest . . . . .	19
utilities . . . . .	20

<b>Index</b>	<b>21</b>
--------------	-----------

---

Maeswrap-package	<i>Bundle of functions for modifying MAESTRA/MAESPA input files, and reading output files.</i>
------------------	--

---

## Description

The main functions are [runmaespa](#) and [maesparunall](#), see their help pages. Functions that read parameters, and modify parameters or whole namelists are [readPAR](#), [replacePAR](#) and [replaceNAMELIST](#). Functions that read output are [readdayflux](#), [readhrflux](#)

## Details

Package:	Maeswrap
Type:	Package
Version:	1.2
Date:	2008-12-03
License:	GPL
LazyLoad:	yes

## Author(s)

Remko Duursma Maintainer: Remko Duursma <[remkoduursma@gmail.com](mailto:remkoduursma@gmail.com)>

## References

See Belinda Medlyn's MAESTRA homepage at: <http://www.bio.mq.edu.au/maestra/>

---

addarrow *Adds an arrow to a plot*

---

### Description

Specify the length and rotation (bearing) of the arrow, and it will be added to the current device. Works for 3D and 2D plots. Different from arrows, where begin and endpoints of the arrow need to be specified.

### Usage

```
addarrow(x0,y0,len,bearing,headlen=0.2*len,
         headangle=25,Nlabel=TRUE,addto=c('rgl','plot','lattice'),...)
```

### Arguments

<code>x0,y0</code>	Begin point coordinates.
<code>len</code>	Length of arrow.
<code>bearing</code>	Direction. See Details.
<code>headlen</code>	Length of arrow head.
<code>headangle</code>	(Half)-Angle of arrow head.
<code>Nlabel</code>	Logical. Add a label 'N' or not.
<code>addto</code>	One of 'rgl', 'plot', or 'lattice', or first letters.
<code>...</code>	Further parameters passed to segments.

### Details

This function is used to add an arrow pointing North on a plot made with `Plotstand`, or `plotuspar`. The bearing is defined as in MAESTRA, where 0 means the X axis points south (and the North arrow thus parallel to the X-axis), with 90 degrees the X axis points West, and so forth.

Because I don't know how to tell if the current device is setup by either `rgl`, `Lattice`, or the base graphics `plot`, the user needs to specify what device to add the arrow to. The default is 'rgl'.

### Value

Adds to the current device. Returns error when no device is open.

### Author(s)

Remko Duursma

**Description**

Functions for running MAESTRA/MAESPA with parameters read from a .csv file. To make multiple runs, create a comma-separated file (.csv) with each row corresponding to a set of parameter values to be used in a simulation. Each column is for a different parameter, with its first entry being the name of the parameter (see Details below). Also needed is a text file with definitions, i.e. how parameter names in the .csv file correspond to parameter names in one of the MAESTRA input files, which file it is, and in which namelist to look (see Details). See Examples below for usage, and how to deal with the results from a batch run. Note that the batch utilities use the executable 'maespa.exe' (runmaespa), or 'maestra.exe' (runmaestra) by default, but you can use others. See Examples on how to set this.

**Usage**

```
runmaespa(whichrow = 1, whichcols = NA, runfile=file.choose(),
          runit = TRUE, executable = "maespa.exe",
          deffile="maeswrapdefinitions.txt", spinup=FALSE, ...)
runmaestra(executable="maestra.exe",...)
maesparunall(whichrows = NA, runfile, whichcols=NA, quiet = FALSE,
             extrafiles="",...)
maestrarunall(executable="maestra.exe",...)
```

**Arguments**

whichrows	Which rows of the runfile to run in the simulations?
whichrow	For one run, which row to run?
runfile	Name of the runfile, needs to be a .csv file, quoted. If left alone, a menu pops up.
quiet	If TRUE, no progress is written to the console.
whichcols	Which columns in the runfile contain parameters to be changed in the simulations?
runit	If FALSE, writes the input files but does not run the model.
executable	Name of the executable.
deffile	Text file with definitions of parameter names in the input files, their locations in files and namelists. See Details.
spinup	For Maespa only : if TRUE, the model is run once, and all final values of soil water content and soil temperature are used to initialize the next run, which is the run that is reported.
extrafiles	Additional files to read and store (See Details).
...	Further parameters passed to runmaespa or readPAR

## Details

Users will typically run `maesparunall`, which runs every row in the comma-separated file denoted by `runfile`. Every column in this runfile is named, with the name corresponding not directly to a parameter or namelist in one of the fortran input files, but rather to an entry in the definition file. This file (argument `deffile`) needs to be in the current workspace. It is a space (or tab)-separated file, with four columns: `'parname'`, `'fileparname'`, `'filename'` and `'namelist'`. An example of this file is provided with this package. Entries of each do not need to be quoted. The namelist entry can be left blank, but it is recommended to provide the namelist where the parameter occurs, for reliability.

The default executable is `maespa.exe` when running `runmaespa`, but others can of course be used. The function `runmaestra` is exactly the same as `runmaespa`, except that the default is to use the executable `'maestra.exe'`. To set the default for the rest of the session, see the last Example below.

Note that these functions should be general, in that they can be used for any Fortran compiled program (`.exe`) that reads parameters from namelists.

By default, `Maes(tra/pa)` reads and saves the `'hrflux.dat'` and `'dayflux.dat'` files, and the `'watbal.dat'` if it exists (i.e., if `Maespa` was used for the simulation). The `'extrafiles'` argument specified additional output files to read and store.

## Value

For `runmaespa`, nothing is returned. Assuming you used `runit=TRUE`, the model is run and output files are written to disk. The function `maesparunall` returns a list with two components: daily and hourly. These components are itself lists with each element a dataframe with the daily or hourly results from the model output. These dataframes are read from the files `dayflux.dat` and `hrflux.dat`.

## Author(s)

Remko Duursma

## Examples

```
## Not run:

#-1. Run the second row of some csv file, using only entries in
# columns 2, 3 and 4. Note that all column names in the .csv
# file must be documented in the definition file, and that the
# definition file must be in the current working directory!
runmaespa(2, runfile="runfiletest.csv", runit=FALSE, whichcols=2:4)

#-2. Run all rows and all parameters from a comma-separated file.
runresults <- maesparunall(runfile="runfiletest.csv")

#-3. Look at first run:
summary(runresults$daily[[1]])

#-4. Summarize across runs with statements like these.
# Sum net photosynthesis across runs:
sapply(runresults$daily, function(dfr)sum(dfr$netPs))
```

```

#-5. Or write all results to disk:
filenames <- paste("MaespaDailyresults", 1:length(runresults$daily), ".txt", sep="")
for(f in 1:length(filenames)){
  write.table(runresults$daily[[f]], filenames[f], sep=" ", row.names=FALSE)
}

#-6. Use different executable:
runmaespa(executable="othermaestra.exe")

#-7. Or set this other executable as the default for the rest of the session,
# so that you only need to set it once (until you restart R anyway).
formals(runmaespa)$executable <- "othermaestra.exe"

#-8. Specify additional output files to read and save:
myrun <- maestrarunall(runfile="myrunfile.csv", extrafiles=c("layflx.dat", "resp.dat"))
# These two files are then available in the 'myrun' list by names 'layflx' and 'resp'.

## End(Not run)

```

---

checkwatbal

*Checks MAESPA water balance*

---

### Description

Adds up fluxes for the soil water balance calculated by MAESPA. Prints the total fluxes (precipitation, transpiration, canopy interception, and so on), and the difference between ingoing and outgoing. This is a debugging/checking tool: if the model has serious problems, there will be a missing sink or source.

### Usage

```
checkwatbal(x = readwatbal(), usemeaset = FALSE)
```

### Arguments

x	A dataframe returned by <a href="#">readwatbal</a> .
usemeaset	Whether to use measured ET from the met file, or simulated (Default).

### Author(s)

Remko Duursma

### See Also

[readwatbal](#)

---

maeswrapdefinitions      *Example Maeswrap definition file*

---

### Description

The MAESTRA/MAESPA wrapper needs a 'definition file', where names of parameters are defined, together with their locations in parameter files and namelists. The 'comment' column is ignored when parsing the definition file.

### Usage

maeswrapdefinitions

### Format

A white space separated dataset (readable with `read.table`).

### References

None.

---

Plotstand      *Plot the stand in 3D*

---

### Description

Reads the MAESTRA trees file, and plots the stand in 3D. Supports all MAESTRA crown shapes except the box shape. Looks for the 'trees.dat' file in the current working directory, unless specified (see Examples). The XY coordinates *must* be present in the 'trees.dat' file. Users will typically only use the 'Plotstand' function.

Optionally reads the crown shape from the 'str.dat' file, and plots the correct crown shape for each species in the stand by reading the multi-species namelists in 'confile.dat' and 'trees.dat'.

The target trees are colored red (unless specified otherwise, see Details), if the 'itargets' is specified in the confile.

### Usage

```
Plotstand(treesfile="trees.dat", strfile="str.dat",
crownshape=c("cone", "ellipsoid", "round",
"halfellipsoid", "paraboloid", "cylinder"),
readstrfiles=TRUE, targethighlight=TRUE,
addNarrow=TRUE, xyaxes=TRUE, labcex=1, axiscex=1,
verbose=FALSE, ...)
Openstand(treesfile="trees.dat")
plottree(crownshape=c("cone", "elipsoid", "ellipsoid", "round",
```

```
"halfellipsoid","paraboloid","cylinder"),
CL=1, CW=1, HCB=1, X=0, Y=0, dbh=0.3, crowncolor="forestgreen",
stemcolor="brown", nz=25, nalpha=25, ...)
  plot3dtriangles(m,...)
coord3dshape(crownshape=c("cone","ellipsoid","halfellipsoid",
"paraboloid","cylinder"),
nz=25, nalpha=25, CL=1, CW=1, x0=0, y0=0, z0=0)
```

### Arguments

treesfile	By default, the 'trees.dat' file in the current dir.
strfile	Not used, yet.
crownshape	Character, "cone","elipsoid","ellipsoid","halfellipsoid","paraboloid","cylinder", or abbreviation.
addNarrow	Logical. Add arrow pointing North?
xyaxes	Logical. Add annotated X and Y axes?
labcex	Relative size of X and Y axis labels.
axiscex	Relative size of X and Y axis annotation.
verbose	If TRUE, writes more info to the screen while plotting.
CL	Crown length (m).
CW	Crown width (m).
readstrfiles	Read the 'str.dat' file(s) to find out crown shape?
targethighlight	Plot the target trees in red?
crowncolor	The color of the tree crowns. Default, obviously, 'forestgreen'.
stemcolor	The color of the tree stems. Default 'brown'.
x0,y0,z0	Coordinates of crown base when calculating 3D coordinates.
HCB	Height of crown base (m).
X,Y	X- and Y-coordinates of tree stem base (m).
dbh	Stem diameter (m). Converted to m if appears to be in cm.
nz	Number of z divisions (increase number to get smoother crowns).
nalpha	Number of angular divisions (increase number to get smoother crowns).
m	3xN matrix (x,y,z coordinates in rows).
...	Further parameters passed (to plottree, or triangles3d).

### Details

Attempts to read indivradx, indivrady, indivhtcrown, indivdiam, and indivhttrunk namelists from the 'trees.dat' file. If any of these fail, the 'all' versions are tried ('allradx', etc.). Although MAESTRA runs fine when no XY coordinates are provided, this plot function crashes. A future implementation will calculate XY coordinates in the same way as MAESTRA.



If the 'strfiles' parameter is set in 'confile.dat' (one str.dat file for each species in the stand), these files are opened and used to set the crown shape by species. Alternatively, you may specify crown-shape as a parameter, and override reading of str.dat by setting readstrfiles=FALSE.

The 'nz' and 'nalpha' arguments specify the 'smoothness' of the crowns: higher values provide more detailed triangulation of the crowns, at the expense of speed.

### Value

An rgl device is opened.

### Author(s)

Remko Duursma

### Examples

```
## Not run:  
  
# Plot the 'trees.dat' file in the current working directory:  
Plotstand()  
  
# Open a dialog box to select a trees.dat file:  
Plotstand(file.choose())  
  
# Save a snapshot to a .png file.  
# Note: make sure to move the 3D plot into view  
# (so that other windows are not blocking it!)  
snapshot3d('myforest.png')  
  
# For publication-quality graphs:  
Plotstand(nz=50, nalpha=50)  
  
## End(Not run)
```

---

plotuspar

*Plot the understorey PAR points*

---

### Description

Reads the 'uspar.dat' file in the current working directory, and plots the incident or absorbed (or diffuse, or direct) PAR at the understorey points. Either produces a plot, or makes a pdf with a plot for each hour of the selected day.

**Usage**

```
plotuspar(what=c("beam","ipar","diff","apar"), dataset=NULL, day=1,
  hour=NA, xlim=NULL, ylim=NULL, makepdf=FALSE,
  outputfile = "apar understorey.pdf", scaleeach=TRUE,
  addNarrow = TRUE)
readuspar()
```

**Arguments**

what	Either 'diff', 'apar', 'ipar', or 'beam' (the default).
dataset	If left alone, reads the uspar dataset.
day	Which day to use, if left alone uses the first day only.
hour	Which hour to plot. If left alone, makes a plot for each hour.
xlim,ylim	X- and Y-axis limits.
makepdf	Logical. If TRUE, produces a pdf in the working directory.
outputfile	Name of the pdf file.
scaleeach	Logical. Rescale grey scale for each plot, or same for all hours?
addNarrow	Logical. Add an arrow pointing North.

**Details**

If addNarrow is TRUE, attempts to read the trees.dat file in the current working directory as well. Prints a warning when this file cannot be opened.

**Value**

A lattice device, or a pdf.

**Author(s)**

Remko Duursma

**Examples**

```
## Not run:

# Plot one hour of the first day, showing incident PAR on understorey:
plotuspar("ipar", day=1, hour=12, makepdf=FALSE)

# Make pdf of the whole day, plotting beam radiation:
plotuspar("beam", day=1, outputfile="beam uspar")

## End(Not run)
```

---

readdayflux	<i>Reads the dayflx.dat output file</i>
-------------	---

---

**Description**

Reads the dayflx.dat MAESTRA/MAESPA output file, returns a clean dataframe. Names of the variables are read from the Columns: line.

**Usage**

```
readdayflux(filename = "dayflx.dat")
```

**Arguments**

filename           Default name of the daily flux file.

**Value**

Returns a dataframe.

**Author(s)**

Remko Duursma

**Examples**

```
## Not run:  
# Read it:  
mysim1 <- readdayflux()  
  
## End(Not run)
```

---

readhrflux	<i>Reads the hrflux.dat MAESTRA/MAESPA output file</i>
------------	--

---

**Description**

Reads the hourly output file (hrflux.dat).

**Usage**

```
readhrflux(filename = "hrflux.dat")
```

**Arguments**

filename            Default name of the (half-)hourly output file.

**Value**

Returns a dataframe.

**Author(s)**

Remko Duursma

**Examples**

```
## Not run:  
  
# Simple as this:  
mysim2 <- readhrflux()  
  
## End(Not run)
```

---

readmet

*Reads the met.dat input file*

---

**Description**

Reads the meteorological input data in the met.dat file.

**Usage**

```
readmet(filename = "met.dat", nlines=-1)
```

**Arguments**

filename            Default name of the met.dat file.  
nlines              Optional, how many lines of the metfile to read?

**Value**

Returns a dataframe.

**Author(s)**

Remko Duursma

## Examples

```
## Not run:  
  
# Simple as pi:  
metdata <- readmet()  
  
## End(Not run)
```

---

readPAR	<i>Reads the value of a parameter in a MAESTRA/MAESPA input file.</i>
---------	---

---

## Description

Reads the value of any parameter in a namelist in one of the MAESTRA/MAESPA input files. Also works for other text files that have the FORTRAN namelist input structure. Optionally specifies in which namelist to look for the parameter.

## Usage

```
readPAR(datfile, parname, namelist = NA, fail=TRUE)
```

## Arguments

datfile	Name of the input file.
parname	Name of the parameter.
namelist	The namelist to look in, otherwise looks in the whole file.
fail	Logical. If TRUE, stops with an error when parameter is not found.

## Value

Either one value, or a vector, depending on how many values are specified for the parameter in the input file.

## Author(s)

Remko Duursma. Thanks to Andreas Ibrom for reporting a bug.

## See Also

[replacePAR](#)

## Examples

```
## Not run:  
# Read the number of trees in the plot:  
readPAR("confile.dat", "notrees", "plot")  
  
# Read the X and Y coordinates:  
readPAR("confile.dat", "xycoords", "xy")  
  
## End(Not run)
```

---

readwatbal

*Reads the watbal.dat MAESPA output file*

---

## Description

Reads the hourly water balance output file ("watbal.dat").

## Usage

```
readwatbal(filename = "watbal.dat")
```

## Arguments

filename           Default name of the (half-)hourly water balance output file.

## Value

Returns a dataframe.

## Author(s)

Remko Duursma

## Examples

```
## Not run:  
  
# Simple as this:  
mywatbalresult <- readwatbal()  
  
# If you want to select the water balance file with a menu:  
readwatbal(file.choose())  
  
## End(Not run)
```

---

replacemetvar	<i>Replace a weather variable</i>
---------------	-----------------------------------

---

### Description

Replaces one (or more) of the weather variables in the met.dat file.

### Usage

```
replacemetvar(replacevar, newvalues, oldmetfile="met.dat",  
newmetfile="metNEW.dat")
```

```
replacemetdata(metdfr, oldmetfile = "met.dat", columns = NA,  
newmetfile = "met.dat", khrs = NA)
```

### Arguments

replacevar	Character. Name(s) of the variable to be replaced.
newvalues	Vector of new values for the weather variable, has to be the same length as the number of records in the met.dat file.
oldmetfile	Default name of the met.dat file that will be modified.
newmetfile	Name of the new met.dat file.
metdfr	Dataframe with met data, to be pasted into a met.dat file.
columns	Optional character string : if the 'Columns' statement in the met.dat file is to be replaced.
khrs	Optional. Number of timesteps per day (by default, read from the met.dat file).

### Value

Returns nothing.

### Author(s)

Remko Duursma

### Examples

```
## Not run:  
  
#:::1::: Replace precipitation with random number between 0 and 2.  
# First find out how many records there are:  
nrecords <- nrow(readmet("met.dat"))  
  
# Make new rain  
newrain <- runif(nrecords, 0, 2)
```

```
# And replace
replacemetvar("PPT",newrain,"met.dat", "newmet.dat")

#:::2::: Replace multiple weather variables.
newtair <- runif(nrecords, 0, 35)

# Have to make a matrix of the variables to be replaced:
newmat <- matrix(cbind(newrain, newtair),ncol=2)

# And give a vector of variable names --in the same order as in the matrix!--.
replacemetvar(c("PPT","TAIR"), newmat, "met.dat", "newmet.dat")

## End(Not run)
```

---

replaceNAMELIST	<i>Replaces a namelist.</i>
-----------------	-----------------------------

---

### Description

Replaces the whole namelist in an input file. All parameters in the namelist must be provided, otherwise MAESTRA/MAESPA will likely crash.

### Usage

```
replaceNAMELIST(namelist = NA,
                datfile = NA,
                vals = list())
```

### Arguments

namelist	Name of the namelist.
datfile	Name of the input file.
vals	A list of values (see example below).

### Value

Nothing is returned. The input file is modified.

### Author(s)

Remko Duursma



**References**

See Belinda Medlyn's MAESTRA homepage at: <http://www.bio.mq.edu.au/maestra/>

**See Also**

[replacePAR](#)

**Examples**

```
## Not run:
replaceNAMELIST(namelist="aerodyn",
  datfile="trees.dat", vals=list(zht=30,zpd=3,z0ht=0.6))

## End(Not run)
```

---

replacePAR	<i>Replace a parameter value</i>
------------	----------------------------------

---

**Description**

Replaces a parameter value in a MAESTRA/MAESPA input file. Also works for a vector of input values.

**Usage**

```
replacePAR(datfile, parname, namelist = NA, newval, noquotes=FALSE)
```

**Arguments**

datfile	Name of the input file.
parname	Name of the parameter to replace the value of.
namelist	Optional. In which namelist to look for the parameter.
newval	New value of the parameter. Can be a single value or a vector, or a matrix (see Details).
noquotes	Logical. If FALSE, does print quotes around character values.

**Details**

If the new parameter value(s) is a vector (or a single value), the values will be placed on a single line in the parameter file. If instead a matrix is provided, each row of the matrix is placed on a separate line.

**WARNING** : The input file is modified. Make sure to backup your original input files!

**Value**

Nothing is returned. The inputfile is modified.

**Author(s)**

Remko Duursma

**See Also**[replaceNAMESLIST](#), [readPAR](#)**Examples**

```
## Not run:
# Replace a parameter with a single value:
replacePAR("trees.dat", "notrees", "plot", newval=100)

# Replace a number of values:
replacePAR("trees.dat", "xycoords", "xy", newval=c(1,1,2,2,3,3))

# Replace a number of values in a different way : this may be useful in
# more complicated programs,
# rr when reading a string from a file (that should be interpreted as a vector).
replacePAR("trees.dat", "xycoords", "xy", newval="1 1 2 2 3 3", noquotes=TRUE)

# Replace a parameter so that the new values are placed on multiple rows.
# Useful for tree namelists with multiple dates and multiple trees
# (where each row corresponds to a tree, and each column to a date.)
m <- matrix(c(1,2,3,4,5,6,7,8,9), nrow=3, byrow=TRUE)
replacePAR("trees.dat", "indivlarea", "values", newval=m)

## End(Not run)
```

---

revchar

*Reverse a character string.*


---

**Description**

Reverses the characters in a character string, unless a vector is supplied, in which case reverses the element of the vector.

**Usage**

```
revchar(x)
```

**Arguments**

x                    A character vector (typically of length 1).

**Details**

When a character vector of length > 1 is provided, reverses the elements of the vector, not the characters itself.

**Value**

A vector.

**Author(s)**

Remko Duursma

**References**

None.

**See Also**

[rev.default](#), [substr](#), [strsplit](#)

**Examples**

```
## Not run:  
# Take a substring, counting from the end:  
substrfromend <- function(x,start,stop)revchar(substr(revchar(x),start,stop))  
substrfromend('filename.txt', 1,3)  
  
# Check if a word is a palindrome:  
s <- 'saippuakivikauppias'  
s == revchar(s)  
  
# A semordnilap:  
cat('I am so stressed, I need to eat', revchar('stressed'),'\\n')  
  
## End(Not run)
```

---

runfiletest

*Example Maeswrap run datafile.*

---

**Description**

Example of a 'run dataset', where each row corresponds to a simulation with parameters set by the column names. The wrapper looks for the column names in the definition file, and sets the parameters based on information in that file.

**Usage**

```
runfiletest
```

**Format**

A comma-separated file.

**References**

None.

---

utilities

*Utility functions*

---

**Description**

Several simple functions for handling strings etc.

**Usage**

```
delempy(x)
maybeQuote(x)
mayberead(...)
printme(x, SEP=" ")
trim(str)
```

**Arguments**

x	A string, or a character vector
str	Also a string
...	A file name
SEP	Separator in printing the namelist

**Details**

`delempy` deletes elements of a vector that are empty strings. `maybeQuote` returns the quoted string unless the argument can be successfully converted to numeric. `printme` is a utility function for printing namelists. `trim` deletes leading and trailing whitespace in a character string.

**Author(s)**

Remko Duursma

**See Also**

[readPAR](#)

# Index

- \*Topic **datasets**
  - maeswrapdefinitions, 7
  - runfiletest, 19
- \*Topic **methods**
  - revchar, 18
- \*Topic **misc**
  - checkwatbal, 6
- \*Topic **package**
  - Maeswrap-package, 2
- \*Topic **utilities**
  - addarrow, 3
  - BatchUtilities, 4
  - Plotstand, 7
  - plotuspar, 9
  - readdayflux, 11
  - readhrflux, 11
  - readmet, 12
  - readPAR, 13
  - readwatbal, 14
  - replacemetvar, 15
  - replaceNAMELIST, 16
  - replacePAR, 17
  - utilities, 20
- addarrow, 3
- BatchUtilities, 4
- checkwatbal, 6
- coord3dshape (Plotstand), 7
- delempty (utilities), 20
- maesparunall, 2
- maesparunall (BatchUtilities), 4
- maestrarunall (BatchUtilities), 4
- Maeswrap (Maeswrap-package), 2
- Maeswrap-package, 2
- maeswrapdefinitions, 7
- maybeQuote (utilities), 20
- mayberead (utilities), 20
- Openstand (Plotstand), 7
- plot3dtriangles (Plotstand), 7
- Plotstand, 7
- plottree (Plotstand), 7
- plotuspar, 9
- printme (utilities), 20
- readdayflux, 2, 11
- readhrflux, 2, 11
- readmet, 12
- readPAR, 2, 13, 18, 20
- readuspar (plotuspar), 9
- readwatbal, 6, 14
- replacemetdata (replacemetvar), 15
- replacemetvar, 15
- replaceNAMELIST, 2, 16, 18
- replacePAR, 2, 13, 17, 17
- rev.default, 19
- revchar, 18
- runfiletest, 19
- runmaespa, 2
- runmaespa (BatchUtilities), 4
- runmaestra (BatchUtilities), 4
- strsplit, 19
- substr, 19
- trim (utilities), 20
- utilities, 20