Package 'YplantQMC'

July 2, 2014

Type Package

Title Plant architectural analysis with Yplant and QuasiMC.

Version 0.5

Date 2013-03-01

Author Remko Duursma. QuasiMC by Mik Cieslak. Uses code by Robert Pearcy (Yplant) and Belinda Medlyn (MAESTRA).

Depends R (>= 2.10), rgl, geometry, LeafAngle, devtools

Suggests maps,gplots,ypaddon

Maintainer Remko Duursma <remkoduursma@gmail.com>

Description An R implementation of Yplant, combined with the QuasiMC raytracer. Calculate radiation absorption, transmission and scattering, photosynthesis and transpiration of virtual 3D plants.

License GPL-2

LazyData yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2013-03-03 12:24:11

R topics documented:

`constructplant` *Construct a 3D plant*

Description

Read in legacy-style Yplant input files into a special object, to be used in any analysis in YplantQMC.

The function constructs an object of class `plant3d`, based on Yplant input files (.p and .l/.lf). Various methods exist for `plant3d` objects, in particular `plot.plant3d` and `summary.plant3d`.

For batch analyses, the function `readplantlist` reads a number of files at a time, and stores the results in a special list (of class `plant3dlist`).

Three plants are provided with `YplantQMC` (and automatically loaded) `:toona`, `pilularis` and `sugarmaple`. See `plantexamples`.

To learn about the format of P and L files, read the detailed account on the Prometheus wiki (Pearcy, Falster & Duursma 2011): `http://goo.gl/Hmyv6`.

Usage

```
constructplant(pfile = NULL, lfile = NULL, qfile = NULL, multiplier=1.0,
X0 = 0, Y0 = 0, Z0 = 0, warn=FALSE, quiet=FALSE)
readplantlist(pfiles=NA, lfiles=NA, lpk="leafplantkey.txt", multiplier=1)
```

Arguments

pfile,lfile	Name of .p and .l (or .lf) file.
qfile	Optionally, instead of a pfile, a Q file format. See Details.
multiplier	Multiplies length dimensions, e.g. to change units.
X0,Y0,Z0	New x,y,z coordinate of the stem base.
warn	If TRUE, writes warnings of minor issues with P file format.
quiet	If TRUE, no messages are ever shown.
pfiles,lfiles	Vectors of .p files and .l files.
lpk	Optionally, a 'leafplantkey' file. See Details.

Details

For legacy Yplant users (Pearcy and Yang 1996, see http://goo.gl/Hmyv6), you will find that constructplant is much more robust with respect to malformed input files. It will also attempt to write error messages when things go wrong.

The **Q file** format is an alternative to .P files, and is much easier to use if the virtual plant does not have stem sections. There are seven columns:

X,Y,Z Coordinates of the leaf base

ang,az Angle and azimuth of the normal to the leaf surface

or Orientation (azimuth angle) of the midrib

L Leaf length

The file is space-delimited (such as the output of write.table), and includes column headers (exactly named as above).

The **leafplantkey** file is a convenient way to organize a large number of plant files. This is a simple comma-separated text file without headers. The order is pfile,lfile (without quotes). For example, a "leafplantkey.txt" file may look like this:

```
acaflo1.p,acaflo.l
acaflo2.p,acaflo.l
acaflo3.p,acaflo.l
acamyr1.p,acamyr.l
acamyr2.p,acamyr.l
acamyr3.p,acamyr.l
acasua1.p,acasua.l
acasua2.p,acasua.l
acasua4.p,acasua.l
acasuaR02.p,acasua.l
acasuaR05.p,acasua.l
acasuaR09.p,acasua.l
```

Value

In the case of `constructplant`, an object of class `plant3d`. For `readplantlist`, an object of class `plant3dlist` (which is simply a list of objects as generated by `constructplant` to ease batch analyses).

Author(s)

Remko Duursma

See Also

`plot.plant3d`,`readp`

Examples

```
## Not run:
# Read one plant:
myplant <- constructplant("sompfile.p","somelfile.l")

# Pfile was in cm - should be in mm. Multiply all length dimensions by 10.
myplant <- constructplant("sompfile.p","somelfile.l", multiplier=10)

# Read a couple of plants.
myplants <- constructplant(pfiles=c("plant1.p","plant2.p"), lfiles=rep("leaf.l",2))


## End(Not run)
```

`crownhull` *Calculates and plots the convex hull around the plant crown*

Description

This function finds the convex hull (and its surface area and volume) spanning the leaves of the 3D plant, using all coordinates of the leaf edges. The result is smallest set of x, y, z points that defines the convex hull, that is, the polyhedral surface that contains all other points, and is convex.

The implementation is a wrapper for the 'convhulln' function in the from package 'geometry'.

Usage

```
crownhull(xyz, plotit = TRUE, alpha = 0.8)
```

Arguments

xyz	An object of class 'plant3d', or a matrix with three columns (xyz coordinates).
plotit	Logical. If FALSE, returns only volume and surface area.
alpha	Transparency (0-1).

Details

Optionaly, uses the 'rgl' package (see plot3d), to add a plot of the hull to the current (rgl) device. Opens a new device if none is currently open. Uses the non-visible function triangles3d to plot the hull, see details there.

The convex hull is calculated with the qhull algorithm, see convhulln and references therein.

Value

A list with components 'crownvolume' and 'crownsurface', giving the volume and surface of the convex hull.

Author(s)

Remko Duursma

References

www.qhull.org,

Duursma, R.A., D.S. Falster, F. Valladares, F.J. Sterck, R.W. Pearcy, C.H. Lusk, K.M. Sendall, M. Nordenstahl, N.C. Houter, B.J. Atwell, N. Kelly, J.W.G. Kelly, M. Liberloo, D.T. Tissue, B.E. Medlyn and D.S. Ellsworth. 2012. Light interception efficiency explained by two simple variables: a test using a diversity of small- to medium-sized woody plants. New Phytologist. 193:397-408.

Examples

```
# Toona example (plant included in package).
crownhull(toona)


# Some xyz data:
coords <- matrix(runif(300,0,1),ncol=3)
library(rgl)
plot3d(coords, col="blue", size=3, axes=FALSE, box=FALSE, xlab="", ylab="", zlab="")
crownhull(coords)
```

examplehemi *Two example hemiphotos in YplantQMC format*

Description

A canopy with a small gap, and a canopy with a large gap. Both are objects of class yphemi, which is normally constructed with the function setHemi.

Source

Thanks to Daniel Falster and Bob Pearcy.

Examples

```
## Not run:
# Small gap
plot(smallgap)

# Large gap, with solar path in June in the south of france.
southfrance <- setLocation(44)
juneday <- setMet(southfrance, month=6, day=21)
plot(largegap, juneday)

## End(Not run)
```

Farquhar *Farquhar-Ball-Berry coupled leaf gas exchange model*

Description

Coupled photosynthesis-stomatal conductance model. A full implementation of the photosynthesis model by Farquhar et al. (1980), as described by Medlyn et al. (2002) (i.e., following their notation), coupled with a few choices of Ball-Berry type stomatal conductance models. The default is that of Medlyn et al. (2011).

Many parameters can be set in this model, including all temperature response parameters and stomatal conductance parameters. See Details for more information.

Usage

```
Farquhar(PAR, Tair, Ca, VPD, RH = 0, Patm = 101, SWP = 0, Vcmax, Jmax, Rd0, G1, ...)
```

Arguments

PAR	Photosynthetically active radiation (mu mol m-2 s-1).
Tair	Air temperature (deg C)
Ca	Atmospheric CO2 (ppm)
VPD	Vapor pressure deficit (kPa)
RH	Relative humidity (ignored unless MODELGS = 2)
Patm	Atmospheric pressure (kPa)
SWP	Soil water potential (MPa)
Vcmax	Required.
Jmax	Required.
Rd0	Dark respiration at 25 deg C (required).
G1	Slope parameter in stomatal conductance model.
...	Other parameters can be set : see Details.

Details

The minimum required parameters are: Vcmax, Jmax, Rd0 and G1. However, the model contains many other parameters, each with their own default value. To override the default value, for example:

```
# Use Default values:
Farquhar(PAR=1000, Tair=20, Ca=380, VPD=1, Vcmax=100, Jmax=150, Rd0=1)

# Change the shape of the light response curve:
Farquhar(PAR=1000, Tair=20, Ca=380, VPD=1, Vcmax=100, Jmax=150, Rd0=1, THETA=0.8)
```

Below is a list and explanation of all the parameters and their default values.

```
# Respiration parameters
Q10F = 0.67  # logarithm of the Q10 (Equation for respiration :
 # RESP =  RD0 * EXP(Q10F * (TLEAF-RTEMP)/10) * DAYRESP
RTEMP = 25       # Reference temperature (T at which RD0 was measured)
DAYRESP = 1.0    # Respiration in the light as fraction of that in the dark.
TBELOW = -100.0  # No respiration occurs below this temperature (degC).

# Stomatal conductance parameters
MODELGS = 4      # model : 4 = Medlyn et al. 2011; 6 = Tuzet et al. 2003.
EMAXLEAF = 999   # Only used when considering soil water stress and MODELGS is not 6.
KTOT = 2         # Leaf-specific hydraulic conductance (mmol m-2 s-1 MPa-1)

G0 = 0.03        # Stomatal leakiness (gs when photosynthesis is zero).

D0L = 5          # Parameter for the Leuning model (MODELGS=3)
GAMMA = 0        # Gamma for all Ball-Berry type models
G1 = 7           # Parameter for all Ball-Berry type models
```

```
GK = 0.3       # Parameter for three-parameter Medlyn et al. 2011 model (MODELGS=5)

SF = 3.2         # Tuzet model parameters (MODELGS=6)
PSIV = -1.9

# Light-response parameters of electron transport rate.
THETA = 0.4      # Shape parameter of the non-rectangular hyperbola.
AJQ = 0.324      # Quantum yield of electron transport.
HMSHAPE = 0.999  # Shape of the hyperbolic minimum function (no need to change)

# Temperature response parameters.
# Parameters for Jmax.
EAVJ = 37259     # Ha in Medlyn et al. (2002)
EDVJ = 200000    # Hd in Medlyn et al. (2002)
DELSJ = 640.02   # DELTAS in Medlyn et al. (2002)

# Parameters for Vcmax.
EAVC = 47590     # Ha in Medlyn et al. (2002)
EDVC = 0.0       # Hd in Medlyn et al. (2002)
DELSC = 0.0      # DELTAS in Medlyn et al. (2002)
```

Note

The Farquhar function is really just a wrapper for the photosyn function in the package GasExchangeR. In that package, the code is borrowed from the MAESTRA model.

Author(s)

Remko Duursma. Original implementation in FORTRAN by Belinda Medlyn (in the Maestra model). See : http://bio.mq.edu.au/maestra/

References

Farquhar, G.D., S. Caemmerer and J.A. Berry. 1980. A biochemical model of photosynthetic CO2 assimilation in leaves of C3 species. Planta. 149:78-90.

Medlyn, B.E., E. Dreyer, D. Ellsworth, M. Forstreuter, P.C. Harley, M.U.F. Kirschbaum, X. Le Roux, P. Montpied, J. Strassemeyer, A. Walcroft, K. Wang and D. Loustau. 2002. Temperature response of parameters of a biochemically based model of photosynthesis. II. A review of experimental data. Plant Cell and Environment. 25:1167-1179.

Medlyn, B.E., R.A. Duursma, D. Eamus, D.S. Ellsworth, I.C. Prentice, C.V.M. Barton, K.Y. Crous, P. De Angelis, M. Freeman and L. Wingate. 2011. Reconciling the optimal and empirical approaches to modelling stomatal conductance. Global Change Biology. 17:2134-2144.

See Also

lightresponse,setPhy

Examples

```
# A ypphy object, using the coupled Farquhar model.
eucphy <- setPhy("Farquhar", leafpars=list(Vcmax=50, Jmax=100, G1=8, G0=0.01, Rd0=1))

# The 'print' method reminds you what it is:
eucphy
```

`getangles` *Read angles from a .p file.*

Description

Reads leaf angles (angle, orientation or azimuth) from a plant file (a Yplant input file with extension `.p`, known as a `pfile`).

Usage

```
getangles(plant, whichangle='An.3')
```

Arguments

`plant`	An object of class `plant3d`, or the name of a `pfile`.
`whichangle`	The name of the angle, in quotes (see `modifypfile`).

Details

If the leaf angle is returned (An.3, the default), all angles are converted so that they are between 0 and 90 degrees. A warning is printed when any angle > 360, which may indicate problems in the data (this is uncommon).

Other angles may be read, see `modifypfile` for a list of the angles in a pfile.

Value

A vector of angles (in degrees).

Author(s)

Remko Duursma

Examples

```
## Not run:
# Two options:
# Get leaf angles from a pfile
ang <- getangles("someplant.p")

# Or from a constructed plant:
myplant <- constructplant("someplant.p","someleaf.l")
ang <- getangles(myplant)

## End(Not run)
```

---

getR *Get crown radius of a plant*

Description

Calculates the crown radius of a plant, given the x,y,z coordinates of the leaves. See Details. Typically not invoked by user, but used by summary.plant3d to estimate crown width.

Usage

```
getR(xyz)
```

Arguments

xyz	A matrix or dataframe of xyz coordinates.

Details

The crown radius is the distance from the centre of the plant to the furthest leaf in that direction. This distance is found for four directions (corresponding to the four quadrants). The centre of the plant is found from the average x,y coordinate (and may thus differ from the 'stem' location).

Value

Returns only the mean crown radius of the four quadrants.

Author(s)

Remko Duursma

See Also

summary.plant3d,crownhull,Silhouette

installQuasiMC *Download and install the QuasiMC executable*

Description

Downloads the Windows/Mac executable necessary for running runYplant and YplantDay.

QuasiMC is developed by Cieslak et al. (2008), which is part of the Virtual Laboratory plant modelling software created at the University of Calgary (http://www.algorithmicbotany.org).

Both files are placed in the directory (for windows, c:/QuasiMC), which is created if it does not exist already.

This function can also be used to update an existing installation.

Usage

```
installQuasiMC(proxy = FALSE)
```

Arguments

proxy	Use TRUE if connecting through a proxy.

Author(s)

QuasiMC by Mik Cieslak, R installation code by Remko Duursma.

References

Cieslak, M., C. Lemieux, J. Hanan and P. Prusinkiewicz. 2008. Quasi-Monte Carlo simulation of the light environment of plants. Functional Plant Biology. 35:837-849.

leafdispersion *Leaf dispersion of 3D plants*

Description

This function calculates the leaf dispersion for 3D plants, following Duursma et al. (2012).

The method is based on the mean distance to k nearest neighbors in 3D. The function 'leafdispersion' computes this observed mean distance (Ok) for a plant (an object of class plant3d), as well as for a square box with randomly distributed leaves at the same leaf area density.

Usage

```
leafdispersion(plant, kneighbors = 5, nreplicate = 10, nleaves=NA, crownvol=NA)
```

Arguments

plant	An object of class 'plant3d'.
kneighbors	Number of neighbors to be used.
nreplicate	For the random distribution, the number of replicates to simulate.
crownvol,nleaves	Crown volume and number of leaves - optional. If not provided, they are calculated from the 'plant' object.

Value

A list with the following components:

Ok Observed distance to k nearest neighbors

Ek_noedge Expected distance to k nearest neighbors, for random distribution; no edge correction.

Ek_edge As above, but with an edge correction

Ek_edgeSD Standard deviation among replicates of Ek_edge

kneighbors Number of neighbors for distance calculation

disp_edge Edge-corrected leaf dispersion (as in Duursma et al. 2012).

disp_noedge Non edge-corrected leaf dispersion

Author(s)

Remko Duursma

References

Duursma, R.A., D.S. Falster, F. Valladares, F.J. Sterck, R.W. Pearcy, C.H. Lusk, K.M. Sendall, M. Nordenstahl, N.C. Houter, B.J. Atwell, N. Kelly, J.W.G. Kelly, M. Liberloo, D.T. Tissue, B.E. Medlyn and D.S. Ellsworth. 2012. Light interception efficiency explained by two simple variables: a test using a diversity of small- to medium-sized woody plants. New Phytologist. 193:397-408.

Examples

```
# Leafdispersion for the Toona plant
leafdispersion(toona)
```

`lightresponse` *Non-rectangular hyperbola*

Description

A simple light response function that predicts leaf photosynthesis from absorbed PAR.

Usage

```
lightresponse(PAR, Amax, phi, theta, Rd, ...)
```

Arguments

`PAR`	Photosynthetically active radiation (mu mol m-2 s-1)
`Amax`	Maximum assimilation rate (the asymptote) (mu mol m-2 s-1)
`phi`	Quantum yield (slope at PAR = 0) (mol mol-1)
`theta`	Shape of light response curve (0 = rectangular hyperbola, 1 = 'Blackman' response).
`Rd`	Dark respiration (**positive** value).
`...`	Further arguments are ignored.

Value

Returns a dataframe with :

A Net assimilation rate (mu mol m-2 s-1)

Author(s)

Remko Duursma

See Also

`setPhy`

makereport *Make a PDF report of several YplantQMC objects*

Description

Produce a report containing standard graphs and summaries of YplantQMC objects. See Details for usage.

Usage

```
makereport(plant = NULL,
phy = NULL,
met = NULL,
hemi = NULL,
ypsim = NULL,
filename = NA)
```

Arguments

plant	An object of class 'plant3d', see constructplant
phy	An object of class 'ypphy', see setPhy
hemi	An object of class 'yphemi', see setHemi
met	An object of class 'ypmet', see setMet
ypsim	An object of class 'yplantsim', see YplantDay
filename	Optional, the name of the output file

Details

This function produces a number of standard plots and prints of five different YplantQMC objects. The plots cannot at the moment not be customized; please see below for the component functions that generate the plots (these usually have more options for customization). Or, modify the code of makereport as you see fit.

To create a report, simply use this command:

```
makereport(plant=myplant, met=asunnyday,
phy=eucleaf, hemi=mycanopy, ypsim=eucsim1)
```

Where myplant, asunnyday, eucleaf, mycanopy and eucsim1 are objects that you have already generated. The function is flexible : you can generate a report on a subset of the objects, for example only on the hemiphoto and the plant:

```
makereport(plant=myplant, hemi=spruceforest)
```

You may want to check out the following functions, which are used in makereport:

viewplot Produces a three-panel plot with side and top views of the plant.

summary.plant3d Summarizes a plant

setHemi Reads a hemiphoto - also describes the plot function.

fitdistribution Fits (and plots) a leaf angle distribution.

setMet Constructs (and plots) a weather object.

plot.leaffile Plots a leaf.

YplantDay A daily Yplant simulation (and a standard plot).

Value

A PDF is generated in the current working directory.

Note

Note that makereport might fail if you have a PDF open with the same name (as you may have generated this report once before on the same day). Make sure to close the PDF before running makereport. If a PDF is generated that can't be opened, use this command:

```
dev.off()
```

And try again.

Author(s)

Remko Duursma

See Also

plot.plant3d,viewplot,YplantDay

---

makeStand *Make a stand of virtual plants*

Description

Make a stand of plants, for use in runYplant, and for visualization. See the example below to get started. Support for runYplant is somewhat experimental, and YplantDay is not supported yet. Proceed at your own risk.

Usage

```
makeStand(plants = list(), xyz = data.frame(x = 0, y = 0, z = 0), plotbox = NULL)
```

Arguments

plants	List of plants to be placed in the stand.
xyz	Data frame (or matrix) with x,y,z locations of the stem positions of the plants.
plotbox	Optional. Plot boundary, used for scaling-up purposes.

Details

The xyz argument must be a dataframe or matrix with three columns, and it is assumed to be in the order X,Y,Z.

The plotbox argument is optional, if it is not provided the plot boundary will be as a rectangle that just fits around the projected crown area. In some cases, the base of the stem can thus fall outside the plot boundary. For now, the plot boundary is only used to calculate the leaf area index, which has no bearing on any simulation results.

Value

An object of class stand3d, methods exist for print, plot, runYplant. And soon, YplantDay.

Author(s)

Remko Duursma

Examples

```
## Not run:

# Make a stand consisting of three 'toona' plants.
toonastand <- makeStand(list(toona,toona,toona),
                        xyz=data.frame(x=c(0,200,100),
                                       y=c(50,50,300),
                                       z=c(0,0,0)))

# The print method shows a very short summary:
toonastand

# Plot the stand
plot(toonastand)


## End(Not run)
```

ModifyPfiles *Modify Yplant input files.*

Description

Reads a plant file (a Yplant input file with extension .p, known as a pfile), and writes a new pfile, by modifying angles, internode lengths, or segment diameters.

Usage

```
modifypfile(pfile = NA, whichvar = NA, outputfile = "tmp.p",
newvalues = NULL)

replaceangles(whichangle="An.3", pfile=NA, outputfile = NA,
newangles = NULL, distobj = NULL)

changeinternodes(pfile=NA,outputfile = "tmp.p",method=c("perc","constant"),
changeperc=50, consvalue = NA)
```

Arguments

whichangle	By default the leaf angle, or one of the other angles (see Details).
pfile	Name of the .p file.
outputfile	Name of the new file.
distobj	An object of class angledist, that is a leaf angle distribution (See examples).
newangles	A vector of angles to be used in the new p file (optional, in stead of distobj).
whichvar	Name of variable in p file to modify (see readp for list of variables.
newvalues	Vector of new values for the variable that is to be replaced.
method	for changeinternodes, a percentage change ("perc") or a new constant value ("constant").
changeperc	If method="perc", change the internodes by this percentage of their orginal value.
consvalue	If method="constant", change all internodes to this constant value.

Details

The function 'modifypfile' is a general function that can be used to modify any variable in a pfile. The following variables can be changed in a pfile :

Az, An Azimuth and angle of stem sections

Or Orientation angle of the midrib of the leaf

Az.1, An.1 Branch sections

Az.2, An.2 Petioles

Az.3, An.3 Leaves

The function replaceangles is a specialized version for replacing leaf angles (and do some specific error checking, or sampling from leaf angle distributions). The function changeinternodes is a special function for lengths of woody segments (technically not quite the same as internodes!).

For replaceangles, an object of class angledist can be constructed with the angledist function in package LeafAngle (See Example below) or from fitting to a sample with fitdistribution. If angles are provided but the vector is too short, the vector will be sampled with replacement.

Value

A new `pfile` is created, by default with the name "tmp.p", unless the argument `outputfile` is set.

Author(s)

Remko Duursma

See Also

`readp`, `fitdistribution`, `readl`, `constructplant`

Examples

```
## Not run:

# Replace angles by sampling from an ellipsoidal distribution:
mydist <- angledist("ellipsoid", distpars=0.7)
replaceangles(pfile="someplant.p", distobj=mydist)

# Make constant angles:
replaceangles(pfile="someplant.p", newangles=45)

# Change new file name:
replaceangles(pfile="someplant.p", outputfile="someplant 45degrees.p", newangles=45)

# Change petiole orientation, choose pfile with dialog box:
replaceangles("Az.2", newangles=runif(300, 0, 360))

# Modify various variables in a pfile, until we end up with an artificial plant,
# useful for testing.
# Order of changes, in this case (although it does not matter!):
# Leaf azimuth, leaf orientation, leaf angle, petiole length, petiole angle.
modifypfile("originalplant.p", whichvar="Az.3", outputfile="testplant.p", newvalues=45)
modifypfile("testplant.p", whichvar="Or", outputfile="testplant.p", newvalues=45)
modifypfile("testplant.p", whichvar="An.3", outputfile="testplant.p", newvalues=-45)
modifypfile("testplant.p", whichvar="L.2", outputfile="testplant.p", newvalues=10)
modifypfile("testplant.p", whichvar="An.2", outputfile="testplant.p", newvalues=45)



## End(Not run)
```

`plantexamples` *Three example 3D plants*

Description

Three virtual plant, in YplantQMC format. Objects of this kind would normally be produced with constructplant.

Source

Plants provided by Jeff Kelly and Kerrie Sendall.

Examples

```
## Not run:
# A sugar maple (Acer saccharum)
plot(sugarmaple)

# A blackbutt (Eucalyptus pilularis)
plot(pilularis)

# Australian red cedar (Toona australis)
plot(toona)

## End(Not run)
```

---

plot.leaffile *Plots a Yplant leaf file (a file with extension .l or .lf) in 2D.*

Description

Produces a plot of a Yplant leaf file, read in using readl.

Usage

```
## S3 method for class 'leaffile'
plot(x, nleaf=1, edgepoints = TRUE, edgecex = 0.8, ...)
```

Arguments

x	Object of class 'leaffile'.
nleaf	Which leaf to plot in the leaf file (if more than one leaf available in the file).
edgepoints	Logical. If TRUE, plots dots on the leaf edge coordinates.
edgecex	If edgepoint=TRUE, cex (i.e. size) of the leaf edge dots.
...	Further parameters passed to plot.default.

Author(s)

Remko Duursma

See Also

readl

Examples

```
## Not run:

# Read and plot a leaf in one go, select a leaf from a menu.
plot(readl())

# Make a pdf of all leaf files in the current working directory:
leaffiles <- list.files(pattern="\\.l$", ignore.case=TRUE)
pdf("Leaf files.pdf", onefile=TRUE)
for(i in 1:length(leaffiles))plot(readl(leaffiles[i]))
dev.off()



## End(Not run)
```

---

plot.plant3d *Plots a plant in 3D*

Description

Based on a plant constructed with constructplant, plots the plant in 3D using the rgl package. Optionally adds the convex hull to the plot.

Usage

```
## S3 method for class 'plant3d'
plot(x,
noleaves=FALSE,
squarewidth=250,
addcrownhull=FALSE,
hullalpha=0.4,
leaffill=TRUE,
leafoutline=TRUE,
stems=TRUE,
cylinderstems=stems,
leafcolor="forestgreen",
markleaves=NULL,
markcolor="red",
stemcol="black",
branchcol="black",
```

```
    petiolecol="brown",
add=FALSE,
shiftxyz=c(0,0,0),...)
## S3 method for class 'plant3dlist'
plot(x,
whichrows=NA,
png=FALSE,
pngsuff="",
keepopen=!png,
windowrect=c(50,50,800,800),
squarewidth=25,
skipexisting=FALSE,
addfilename=FALSE, ...)
```

Arguments

x	Object of class plant3d or plant3dlist.
noleaves	Logical. If TRUE, leaves are not plotted.
squarewidth	Width of the square to be plotted at the ground surface.
addcrownhull	Logical. If TRUE, a convex hull is added to the plot.
hullalpha	Opacity of the convex hull.
leaffill	Logical. If TRUE, the default, fills in the leaves.
leafoutline	Logical. If TRUE, plots leaf outlines.
stems	Logical. If TRUE, plots the stems (black) and branches .
cylinderstems	Logical. If TRUE, plots the stems as cylinder sections given the diameter in the plant file.
leafcolor	Color of the filled leaves. Can only specify one color.
markleaves,markcolor	A vector of leaf numbers to 'mark', using 'markcolor'.
stemcol,branchcol,petiolecol	Color of the stems, branches, petioles
add	Logical. If TRUE (defaults to FALSE), adds this plant to an existing scene.
shiftxyz	Vector of x,y,z coordinates to shift the plant (see Examples).
whichrows	Which of the plants in the plant3dlist object to plot. Defaults to all.
png	If TRUE, saves a PNG snapshot of every plant plotted.
pngsuff	A suffix to add to the filename (defaults to pfile.PNG).
keepopen	Whether to keep all windows open. Max number of windows is 10.
windowrect	Size of the rgl window (see open3d). Affects PNG quality.
skipexisting	If PNG=TRUE, whether to skip plants that are already saved to disk.
addfilename	If TRUE, places a label of the pfile on the plot (currently poorly placed).
...	Further arguments passed to plot3d in the rgl package.

Details

Some of the hard work, figuring out the X,Y,Z coordinates of the leaf tip and leaf sides, is directly taken from the original Yplant Delphi code (in the non-visible function `madeleafdirection`.

See the function `crownhull` for details of the convex hull algorithm.

Author(s)

Remko Duursma. Robert Pearcy for the Y-plant code.

See Also

`crownhull`

Examples

```
## Not run:
# Construct a plant like this:
# myplant <- constructplant("somepfile.p","aleaf.l")

# Standard plot (using built-in sugar maple):
plot(sugarmaple)

# Add a convex hull.
plot(sugarmaple, addcrownhull=TRUE)

# Plot two plants in one scene (second plant is moved 750mm in the x-direction)
plot(sugarmaple)
plot(sugarmaple, shiftxyz=c(750,0,0), add=TRUE)

# Mark the first 10 leaves in red (i.e. first 10 leaves in P file):
plot(sugarmaple, markleaves=1:10)

# Mark all leaves on the plant that have a leaf angle > 45.
plot(toona, markleaves=which(toona$leafdata$ang > 45))

# Plot the stems (and branches) only:
plot(pilularis, noleaves=TRUE)

# Plot many plants.
# First organize a 'leafplantkey', a comma-separated file that links each pfile to an lfile.
# See the online manual for an example, or the help file for 'constructplant'.
myplants <- readplantlist(lpk="leafplantkey.txt")
plot(myplants, png=TRUE, addfilename=TRUE)


## End(Not run)
```

projectplant *Project plant coordinates onto a viewing plane*

Description

Transform all leaf edge coordinates onto a viewplane coordinate system, with the 'z' axis pointing toward the viewer.

Usage

```
projectplant(plant, azimuth, altitude)
## S3 method for class 'projectedplant3d'
plot(x, silhouette = FALSE, xlim = NULL, ylim = NULL,
leaffill = TRUE, leafcol = "forestgreen", zerocenter = FALSE, xlab = "X", ylab = "Y", ...)
```

Arguments

plant	Object of class 'plant3d' (see constructplant.
azimuth,altitude	Azimuth and altitude from which plant is viewed.
x	Object of class 'projectedplant3d'.
silhouette	Add a 2D convex hull (see Silhouette).
xlim,ylim	Limits for the X and Y axes.
leaffill	If TRUE, fills leaves with green stuff.
leafcol	If leaffill=TRUE, the color of the leaves.
zerocenter	Whether to shift the plant to X=0 and Y=0.
xlab,ylab	Labels for X and Y axes
...	Further parameters passed to plot() (but ignored for print).

Value

Returns an object of class 'projectedplant3d', with components:

leaves	A list of matrices with the coordinates of the leaf edges. Each matrix has columns VX,VY and VZ, which are the viewplane coordinates (with the Z axis pointing toward the viewer).
viewbound	List of min and max values of coordinates (minx, maxx, miny, etc.).
viewangle	Azimuth and altitude used for projection.

Author(s)

Remko Duursma

See Also

STARbar

Examples

```
# View a plant from above.
# The 2D convex hull is also plotted (the 'silhouette'),
# and the area of the hull is printed on the graph.
topview <- projectplant(pilularis, altitude=90, azimuth=180)
plot(topview, leaffil=TRUE, silhouette=TRUE)
```

---

psrdata *Get PSR data.*

Description

Extract Plant Summary Report from a YplantQMC simulation.

Usage

```
psrdata(x, ...)
```

Arguments

x	Object returned by YplantDay
...	Further arguments ignored

Details

Simple wrapper: `x$psrdata` is the same as `psrdata(x)`.

Author(s)

Remko Duursma

randomplant *Generate a plant with randomly distributed leaves*

Description

Generates a plant crown with specified total leaf area or number of leaves, in one of six crown shapes. These include cone, paraboloid, cylinder, and others. Leaves are randomly distributed in the crown shapes.

Usage

```
randomplant(nleaves = 500,
radius = 250,
height = 500,
shape = c("BOX", "CONE", "HELIP", "PARA", "ELIP", "CYL"),
crownbase = 0,
LAD = angledist("ellipsoid", 1),
leaflen = 30,
LA = NULL,
lfile = NULL,
writefile = FALSE,
quiet = FALSE)
```

Arguments

nleaves	Number of leaves, can be left unspecified if leaf area (LA) is given.
radius	Crown radius (mm).
height	Crown height (mm).
shape	One of six pre-specified crown shapes. See Details.
crownbase	Height to crownbase (mm).
LAD	Leaf angle distribution.
leaflen	The leaf length (constant value for all leaves).
LA	Total leaf area, optional (m2)
lfile	Name of the leaf file (see readl), if left unspecified a triangle is used.
writefile	Logical. If TRUE, writes a Q file to disk.
quiet	Logical. If FALSE, no messages are printed to the console.

Details

Six crown shapes are implemented, use one of the following abbreviations for the argument crownshape:

BOX A box shape; radius in X and Y directions assumed to be the same.

CONE A cone shape.

HELIP A half ellipsoid (the top half of a full ellipsoid).

PARA A paraboloid.

ELIP A full ellipsoid.

CYL Cylinder shape.

The leaf angle distribution (LAD) must be specified using the angledist function in the LeafAngle package. To assign a constant leaf angle of 45 degrees, use this command:

```
LAD = angledist("const", 45)
```

To assign a spherical LAD, use this command:

```
LAD = angledist("spherical")
```

Value

Produces a plant in the Q file format, see constructplant for details. If writefile=TRUE, the Q file is written to the current working directory. Otherwise, returns a dataframe that can be used directly in constructplant, see Details.

Note

Currently only random distributions are supported. I have at this time no idea how to produce clumped or uniform distributions in 3D (if you do, let me know).

Author(s)

Remko Duursma. Uses code from Belinda Medlyn (SURFACE subroutine in MAESTRA).

See Also

constructplant

Examples

```
## Not run:
# An ellipsoid shape crown, write a Q file to disk.
# Specify approximate total leaf area.
randomplant(radius=400, height=2000, shape="ELIP",
leaflen=40, LA=1.5, writefile=TRUE)

# Embed the function in a call to 'constructplant', giving a plant in the
# YplantQMC format.
# Because no leaf file is specified, it uses a built-in triangle-shaped leaf.
 coneplant <- constructplant(randomplant(radius=250, height=500, shape="CONE",
 leaflen=25, LA=1))


## End(Not run)
```

readl *Read a Yplant leaf file*

Description

Reads a Yplant leaf file (.l or .lf) into an object of class `leaffile`.

NOTE: if there are more than one leaf type type in the leaffile, it only reads the first one. At the moment, YplantQMC only uses one leaf file for all leaves in the canopy.

Contrary to the original Yplant, `readl` is a bit picky with leaffiles: there must be a point in the leaffile where X=0 (apart from the first and last points), which is used to find the 'midrib', which in turn is used to calculate leaf length.

Usage

```
readl(lfile = NA)
```

Arguments

`lfile`	Name of the leaf file (character string).

Value

An object of class `leaffile`. Currently, only a plot method exists (see `plot.leaffile`).

Author(s)

Remko Duursma

See Also

`plot.leaffile`

readp *Reads the data from a .p file.*

Description

Reads all data of the plant input file (`.p`) into a dataframe.

Usage

```
readp(pfile=NA)
```

Arguments

`pfile`	Name of the p file (a character string).

Value

Dataframe. The column names correspond to the labels in the .p file, but duplicates have a numeric subscript;

N - node number

MN - mother node

ste - stem (1) or branch (2)

Lt - leaf type

Az, Az.1, Az.2 - Azimuth of stem, branch, petiole and leaf.

An, An.1, An.2 - Angle of stem, branch, petiole and leaf.

L, L.1, L.2, L.3 - Length of stem, branch, petiole and leaf.

More detail on the leaf angles:

Az.3 - Leaf azimuth (azimuth angle of the normal to the leaf surface).

Or - Leaf orientation (azimuth angle of the midrib).

An.3 - Leaf angle (angle between normal to the leaf surface and the z-axis).

Author(s)

Remko Duursma

`runYplant` *A single simulation of YplantQMC*

Description

Runs the YplantQMC model for one timestep. Runs the QuasiMC raytracer to estimate absorbed PAR for every leaf on the plant, given diffuse and direct radiation (set by `fbeam`, see below), the position of the sun, and reflectance and transmittance of the foliage material.

Required is a 3D plant object (see `constructplant`). Optionally, a leaf gas exchange model is used (not needed if only light absorption is calculated) to calculate photosynthesis (and optionally, transpiration rate). Also optionally, a hemiphoto object is used to calculate shading by the overstorey canopy.

Output is not as easy to use as the more user-friendly `YplantDay`. If you are only interested in diurnal simulations (and plant totals by timestep), use that function. The `runYplant` function is available for programming purposes (and more advanced custom simulations).

Usage

```
## S3 method for class 'plant3d'
runYplant(x, phy = NULL, hemi = NULL, reldiff = NULL,
reldir = NULL,  altitude = 90, azimuth = 0, fbeam = 1, VPD = 1.5,
PAR0 = 1, PARwhere = c("above", "below"), Ca = 390, Tair = 25,
Patm = 101, reflec = c(0.1, 0.1), transmit = c(0.1, 0.1),
runphoto = TRUE, intern = TRUE,
debug = FALSE, delfiles=TRUE, rewriteplantfile = TRUE, ...)
## S3 method for class 'stand3d'
runYplant(x,...)
```

Arguments

x	An object of class 'plant3d', see constructplant
phy	An object of class 'ypphy', see setPhy
hemi	An object of class 'yphemi', see setHemi
reldiff	Optional. A vector of relative diffuse absorption, same length as number of leaves. See Details.
reldir	Optional. A vector of relative direct absorption, same length as number of leaves. See Details.
altitude,azimuth	Solar altitude and azimuth (degrees).
fbeam	Beam fraction (0-1). If 0, only diffuse interception is calculated, if 1, only direct.
VPD	Vapor pressure deficit (kPa)
PAR0	Incident PAR on a horizontal surface (mu mol m-2 s-1).
PARwhere	If 'above', PAR0 is given as an above-canopy value. If 'below', it is below the canopy. See Details.
Ca	Atmospheric CO2 concentration (ppm).
Tair	Air temperature (deg C).
Patm	Atmospheric pressure (kPa).
reflec	Leaf reflectance (top, bottom of leaf).
transmit	Leaf transmittance (top, bottom of leaf).
runphoto	Whether to run leaf gas exchange model (default TRUE, or FALSE when no phy object given).
intern	If FALSE, returns output of QuasiMC to the console.
debug	If TRUE, opens the QuasiMC debug window (for testing).
delfiles	If TRUE, deletes intermediate files, and QuasiMC in/output files.
rewriteplantfile	If TRUE, writes the plant QuasiMC input file.
...	Further arguments passed to writecfg.

Details

The arguments `intern`, `debug`, `delfiles` and `rewriteplantfile` should not be set by the user, unless you really know what you are doing. These arguments exist for testing, and are used by `YplantDay`.

The arguments `reldiff` and `reldir` can be supplied if they are already known (from a previous simulation, when the solar angle was the same, in particular). If you are not sure, please do not set these arguments!

Value

This returns a dataframe with one row per leaf. The variables included are (PAR is in units mu mol m-2 s-1):

PAR0 Incident PAR on a horizontal surface *above* the canopy

PARinc Incident PAR on a horizontal surface *below* the canopy

PARleaf Absorbed PAR (for each leaf)

PARdir Absorbed direct PAR

PARdiff Absorbed diffuse PAR

reldiff Relative diffuse absorbed PAR (0 - 1)

reldir Relative direct absorbed PAR (0 - 1)

LA Leaf area (mm2)

LAproj Projected leaf area (mm2)

LAsunlit Sunlit leaf area (mm2)

A CO2 assimilation rate (mu mol m-2 s-1)

E Transpiration rate (mmol m-2 s-1)

gs Stomatal conductance (mol m-2 s-1)

A0 CO2 assimilation rate for a horizontal leaf *below* the canopy.

Author(s)

Remko Duursma

See Also

`YplantDay`, `setPhy`, `setHemi`.

Examples

```
## Not run:

# Compare diffuse only to direct only
run_dir <- runYplant(pilularis, fbeam=1, altitude=90, azimuth=0, reflec=0.15, transmit=0.1)
run_diff <- runYplant(pilularis, fbeam=0, reflec=0.15, transmit=0.1)
```

```
# Compare density functions of absorbed PAR by leaf:
plot(density(run_dir$PARleaf, from=0, to=1), xlim=c(0,1), main="", lwd=2, col="blue",
xlab="Absorbed PAR (relative units)")
lines(density(run_diff$PARleaf, from=0, to=1), lwd=2, col="red")
legend("topright",c("Diffuse","Direct"), lwd=2, col=c("red","blue"))

## End(Not run)
```

`setHemi` *Generate a hemiphoto object*

Description

Construct an object that contains information on shading by the canopy, as measured with hemispherical photographs ('hemiphotos'). Reads one of two formats. See YplantQMC website for example files, and the instruction manual for more background.

Usage

```
setHemi(canfile, canopytrans = 0.02)
evalHemi(hemi, altitude = NULL, azimuth = NULL,
met = NULL, degrees = TRUE)
## S3 method for class 'yphemi'
plot(x, met=NULL, sungap=TRUE,
projection=c("iso","flat"), warn=TRUE, bordercol='black', ...)
```

Arguments

`canfile`	A canopy file (see Vignette for format example).
`canopytrans`	Minimum transmission of canopy (when gap fraction = 0).
`hemi`	An object generated with `setHemi`.
`altitude,azimuth`	Viewing altitude and azimuth.
`met`	Optionally, a `ypmet` object.
`degrees`	Whether altitude and azimuth were given in degrees (if FALSE, radians).
`x`	For `plot.yphemi`, a `yphemi` object
`sungap`	If TRUE, and a `ypmet` object is used, plots the sun along the path, with the size relative to the gap fraction
`projection`	If 'iso', each altitude bin has the same width in the plot (Default). If 'flat', projection is as seen from above.
`warn`	If TRUE and sungap=TRUE, warns when gap fractio is very low.
`bordercol`	Color of the grid separating the sky sectors. Use `NA` to omit borders.
`...`	Further arguments passed to plot.default

Author(s)

Remko Duursma

References

For background on hemiphotos : http://en.wikipedia.org/wiki/Hemispherical_photography

See Also

setMet

---

setLocation *Generate a location object*

Description

Construct an object that contains a geographical location given by the latitude, longitude, and (optionally), the longitude of the nearest timezone border. This object is required in setMet.

Usage

```
setLocation(lat = NA, long = 0, tzlong = NA)
```

Arguments

lat	Latitude (degrees, southern hemisphere negative)
long	Longitue (degrees)
tzlong	Optional. Longitude of the nearest timezone border (degrees).

Details

If the longitude of the nearest time zone border (tzlong) is not set, it is assumed that all times are in 'local apparent time' (LAT), where maximum solar altitude is reached at noon. If it is set, it is used to calculate the time offset between clock time and solar time.

If lat and long are not given, a location can be selected from a simple map of the world. The maps package is needed for this utility.

You can also plot the location on a map of the world (see Examples).

Value

An object of class yplocation.

Author(s)

Remko Duursma

See Also

setMet

Examples

```
# Set a location:
sydney <- setLocation(lat=-33.5, long=152, tzlong=150)

## Not run:
# Set a location from a map:
somewhere <- setLocation()

# Plot locations:
plot(sydney)

## End(Not run)
```

---

setMet *Generate a weather object*

Description

To run Yplant, a weather object needs to be constructed, that contains solar position data, radiation, air temperature, and so on. This function generates a daily diurnal weather dataset using a fairly standard weather generator, or constructs the weather object with user-specified data. See Details.

Usage

```
setMet(location=NULL,
metdat=NULL,
year = 2012,
month = NA,
day = NA,
nsteps = 10,
PARday = 22,
AtmTrans = 0.76,
fbeamday = NA,
fbeammethod = c("spitters", "constant"),
Tmin = 10,
Tmax = 25,
VPDmax = NA,
maxlag = 0.1,
Ca = 390,
Patm = 101)
```

Arguments

location	A Yplant location object (class 'yplocation', see setLocation).
metdat	Optionally, a dataframe (or name of CSV file) with standard weather variables.
year	Optional (slight effects on solar path).
month	1-12
day	day of month
nsteps	number of steps (will affect number of simulation steps in YplantDay.
PARday	Total daily PAR on a horizontal surface (mol m-2 d-1).
AtmTrans	Atmospheric transmission.
fbeamday	Daily beam fraction.
fbeammethod	If 'Spitters', uses the Spitters algorithm to estimate fbeam by timestep, otherwise it is constant (and given by fbeamday.
Tmin,Tmax	Daily minimum and maximum temperature (deg C).
VPDmax	Optional. Daily maximum VPD (if not given, estimated from Tmin).
maxlag	Lag of temperature maximum behind solar maximum (fraction of day).
Ca	Atmospheric CO2 concentration (ppm).
Patm	Atmospheric pressure (kPa).

Details

A built-in weather generator simulates the following variables:

altitude,azimuth Position of the sun (degrees).

PAR Photosynthetically active radiation (mu mol m-2 s-1).

fbeam Fraction direct beam of PAR (-).

Tair Air temperature (deg C).

VPD Vapor pressure deficit.

The following two variables are user input, and have no within-day variation:

Ca Atmospheric CO2 concentration (ppm). (Default = 390ppm).

Patm Atmospheric pressure (kPa). (Default = 1.01kPa).

If you are curious about the algorithm, please check the code (type setMet).

To generate a weather dataset, simply use this command:

```
aprilday <- setMet(richmond, nsteps=12, Tmin=9, Tmax=25, month=6, day=21)
```

Where richmond is a Yplant location object, generated with setLocation.

The weather object can be plotted: the following command produces a simple built-in graph of PAR, Tair, VPD and fbeam:

```
plot(aprilday)
```

Alternatively, the user can input a dataframe (or CSV file) that contains the weather variables (or a subset of them). For example,

```
mymet <- data.frame(Tair=20, PAR0=seq(5,1000,length=10), fbeam=0, Ca=400)
```

The names of the variables need to be *exactly* as described above (and are case-sensitive!).

If solar altitude and azimuth are not provided, they will be calculated from the location object. In the case that `fbeam = 0`, though, the solar position has no effect and is ignored, and not calculated.

Value

An object of class 'ypmet', a list with the following components:

dat A dataframe with the weather variables (see Details for a description).

method Either 'generated' (weather generator was used), or 'input' when user provided `metdata`.

daylength in hours

sunset,sunrise in hours

location A Yplant location object (class 'yplocation', see `setLocation`

Author(s)

Remko Duursma. Solar path and diffuse partitioning code borrowed from Maestra (thanks to Belinda Medlyn).

References

For fraction diffuse radiation, uses the 'Spitters algorithm':

Spitters, C.J.T., Toussaint, H.A.J.M., Goudriaan, J., 1986, Separating the diffuse and direct component of global radiation and its implications for modeling canopy photosynthesis. Part I. Components of incoming radiation, Ag. For. Meteorol., 38:217-229.

See Also

`setPhy`,`setLocation`

`setPhy` *Make a leaf physiology object*

Description

Constructs an object of class 'ypphy', which contains a function that calculates leaf photosynthesis and transpiration (and possibly other variables), from weather data (air temperature, humidity, etc.), and absorbed PAR.

Users can write their own leaf gas exchange functions to be included in a physiology object, or use one of two built in functions: the Farquhar model (see `Farquhar`), or a simple non-rectangular light response curve (see `lightresponse`).

Usage

```
setPhy(leafmodel, leafpars = list())

## S3 method for class 'plant3d'
includePhy(object, ...)
## S3 method for class 'plant3dlist'
includePhy(object, phydfr, ...)
```

Arguments

leafmodel	Name of the leaf gas exchange model ('Farquhar', or 'lightresponse', or user-defined).
leafpars	List of parameters that are passed to the leafmodel (and should be arguments of that function).
object	A 'plant3d' object (see constructplant), or a 'plant3dlist' object.
phydfr	A dataframe with leaf parameters, for batch analyses (see Details).
...	Further arguments passed to 'setPhy'

Details

A typical usage of setPhy is :

```
eucphy <- setPhy("Farquhar", leafpars=list(Vcmax=80, Jmax=140, Rd=1, G1=7))
```

This object may be used when running Yplant directly (see YplantDay, or it may be saved into a plant object (which makes it somewhat easier to organize, especially for batch processing). This is achieved with the includePhy function:

```
myplant <- includePhy(myplant, eucphy)
```

To find out whether a plant has a physiology object saved in it, simply type:

```
myplant$phy
```

If there is a physiology object, it will print a summary of its contents, otherwise it is NULL.

For batch analyses, includePhy can set the leaf parameters for a list of plants (as constructed with readplantlist). To do this, construct a dataframe where each row corresponds to a set of parameters for a plant, and the columns include pfile (required, to match the parameters to the plants in the list), leafmodel (required, the name of the leaf model), and further any parameters that can be accepted by the leafmodel (for example, Vcmax or Amax, and so on). Then use this command,

```
myplantlist <- includePhy(myplantlist, leafpardataframe)
```

Value

An object of class 'ypphy'.

Author(s)

Remko Duursma

See Also

Farquhar,lightresponse,makereport

Silhouette *Calculates the area of the 2D convex hull of a projected plant*

Description

to be added

Usage

```
Silhouette(obj, azimuth = NA, altitude = NA)
```

Arguments

obj	Either a plant3d object, or a projectedplant3d object.
azimuth,altitude	Viewing azimuth and altitude (ignored if a projected plant is given as input).

Note

Not usually called by the user. Use instead, STARbar and projectplant. For the latter, see the option silhouette=TRUE.

Author(s)

Remko Duursma

See Also

summary.plant3d,STARbar,projectplant

Examples

```
# Silhouette returns the area of the 2D convex hull (H), and the coordinates of the 2D hull (xyz):
Silhouette(sugarmaple, altitude=0, azimuth=45)
```

STARbar *Calculate displayed and projected leaf areas of a 3D plant*

Description

From a 3D plant, calculates the area of all leaves facing the viewing angle (from a given azimuth and altitude), or 'projected area', and the area of the leaves that is visible ('displayed area'). By default, repeats calculations from many viewing angles to estimate the hemi-spherically averaged STAR, or $\overline{STAR}$.

Usage

```
## S3 method for class 'plant3d'
STARbar(object,
method=c("gridtracer","exact","QuasiMC","slowquasimc"),
integration=c("Turtlesky","Yplant","Turtle244"),
progressbar=TRUE,
returnldr=FALSE,
quiet=FALSE,
npside=50,
silhouette=TRUE,
azimuth=NA,
altitude=NA, ...)
## S3 method for class 'plant3dlist'
STARbar(object,
quiet=FALSE, ...)
```

Arguments

object	An object of class 'plant3d', see constructplant
method	The method to calculate the displayed area. See Details.
integration	The integration method to calculate the average STAR over the hemisphere. See Details.
progressbar	If TRUE (default), displays a graphical progress bar.
npside	For method = "gridtracer", the number of grid cells per side for raytracing.
returnldr	If TRUE, returns a dataframe with results per viewing angle.
quiet	If TRUE, prints no progress to the screen.
silhouette	If TRUE, also calculates the 2D convex hull around the projected plant (see Silhouette).
azimuth, altitude	Azimuth and altitude of the viewing direction (Optional, in Radians).
...	Further arguments are ignored, for now.

Details

This function calculates the displayed area (DA) and projected area (PA) of the entire plant, either over some specified viewing angle(s), or (by default) the hemispherically average values. From these averages, the $\overline{STAR}$ is estimated (see Duursma et al. 2012).

The $\overline{STAR}$ can also be calculated with the `summary.plant3d` function.

There are four methods for the calculation of the displayed and projected area from a given viewing angle:

gridtracer A regular grid is placed over the projected plant. Displayed area is calculated from the number of intersecting grid points, and the grid size. The argument `npside` sets the number of grid points on the shorter side of the projected plant. Slow for large plants, but good estimates are still obtained with low values of `npside`.

exact A polygon method to calculate the exact displayed area. Useful for testing, but *very* slow.

QuasiMC Runs `QuasiMC` over the entire hemisphere - the `QuasiMC` model does the averaging. Relatively slow for small plants, but fast for large plants.

slowquasimc Do not use this method (for testing only). Runs `QuasiMC` once per viewing angle, then averages the results. Painfully slow.

There are three integration methods. Note that if `method = "QuasiMC"`, the integration method is ignored.

Turtlesky Integrates over 58 points, that are placed approx. uniformly over the hemisphere.

Yplant Integrates over 160 angles, distributed over the hemisphere as in the original Yplant. Weighing is applied over these angles to yield the average STAR.

Turtle244 As Turtlesky, but uses 244 points (slow method).

Author(s)

Remko Duursma

References

Duursma, R.A., D.S. Falster, F. Valladares, F.J. Sterck, R.W. Pearcy, C.H. Lusk, K.M. Sendall, M. Nordenstahl, N.C. Houter, B.J. Atwell, N. Kelly, J.W.G. Kelly, M. Liberloo, D.T. Tissue, B.E. Medlyn and D.S. Ellsworth. 2012. Light interception efficiency explained by two simple variables: a test using a diversity of small- to medium-sized woody plants. New Phytologist. 193:397-408.

See Also

`projectplant`,`constructplant`,`Silhouette`, `summary.plant3d`

Examples

```
# Get STARbar for the built-in Toona plant:
# (Settings are fast, not accurate)
STARbar(toona, method="gridtracer", npside=15)
```

```
## Not run:
# For exact STAR, use:
STARbar(toona, method="exact")

# To produce an LDR file (as in the original Yplant), use:
clidstar <- STARbar(toona, method="gridtracer", npside=30, integration="Yplant", returnldr=T)
write.table(clidstar$ldr, "toona.LDR")

## End(Not run)
```

---

summary.plant3d *Summarize 3D plants*

Description

Summarize the 3D plant in various useful ways. Requires an object of class plant3d, made with constructplant.

Usage

```
## S3 method for class 'plant3d'
summary(object, nKErepeat = 10, nsignif = 3, calcSTARbar=FALSE, ...)
## S3 method for class 'plant3dlist'
summary(object, writefile=FALSE, ...)
```

Arguments

object	Object of class 'plant3d' (one 3D plant) or 'plant3dlist' (a list of 3D plants).
nKErepeat	Number of replicates for leafdispersion, see its help page.
nsignif	Number of digits for output (only for printing).
writefile	If TRUE, writes a text file with the summary results in the cur. working dir.
calcSTARbar	If TRUE, also calculates STARbar and adds it to the summary result.
...	Further arguments passed to STARbar.

Details

The summary.plant3d prints a number of plant summary variables. They are also stored in a list. These are the variables that are currently calculated:

```
LA           -  Total leaf area (m2)
meanleafsize -  Mean leaf size (cm2)
nleavesp     -  Number of leaves
leaflen      -  Mean leaf length (cm)
meanleafang  -  Mean leaf angle (deg)
wmeanleafang -  Mean leaf angle weighted by leaf area (deg)
```

```
Xellipsoid   -  Ellipsoidal leaf angle dist. par.
crownvol     -  Crown volume (convex hull) (m3)
crownsurf    -  Crown surface area (convex hull) (m2)
ALAC         -  Crown density (AL/AC) (m2 m-2)
cw           -  Crown width (m)
cl           -  Crown length (m)
htot         -  Total height(m)
cshape       -  Crown shape index (-)
stemsurf     -  Stem + branch surface area (cm2)
stemvol      -  Stem + branch volume (cm3)
stemdiam     -  Stem base diameter (mm)
meanpath     -  Mean pipe length (mm)
sdpath       -  Standard deviation of pipe length (mm)
totlen       -  Total woody segment length (mm)
Ek           -  Expected distance to 5 nearest leaves (no edge corr.)
Ek2          -  Expected distance to 5 nearest leaves (with edge corr.)
Ok           -  Observed distance to 5 nearest leaves
disp         -  Dispersion parameter (no edge corr.)
disp2        -  Dispersion parameter (with edge corr.)
STARbar      -  (Optional, only when calcSTARbar = TRUE).
```

Note that when generating a summary on a `plant3dlist` object, the above information is written to an outputfile. The outputfile is tab-delimited, and has the name 'Plant summaries-YYYY-MM-DD.txt', using the current date.

The following functions are called to calculate some of the summary variables: `leafdispersion` for Ek,Ek2,Ok,disp,disp2; `pathlen` (hidden function) for meanpath,sdpath,totlen; the `fitdistribution` function in the `LeafAngle` package for Xellipsoid; `getR` for cw; and `crownhull` for crownsurf, crownvol and ALAC. The remainder of the variables are trivial calculations from the plant input file (the .p file) or the leaf file.

Optionally, the $\overline{STAR}$ is calculated (when `calcSTARbar = TRUE`), by calling `STARbar`. See its help page for full details. Note that all options for `STARbar` can also be set with the summary function (see Examples).

Value

A list with components described above. See also the Examples below.

Author(s)

Remko Duursma

See Also

`crownhull`, `leafdispersion`, `getR`

Examples

```
# Print summary (use built-in Toona plant):
```

```
summary(toona)

# Or save summary as a list, access single values:
plantsumm <- summary(toona)
plantsumm$meanpath  # mean path length from soil to leaf
# See table above for names of single variables.

## Not run:
# Summary on a plant3dlist ('myplants' is constructed with 'readplantlist').
summary(myplants, writefile=TRUE)

# Also calculate STARbar (with the exact method).
summary(myplant, calcSTARbar=TRUE, method="exact")

## End(Not run)
```

`turtle` *A turtle sky with 58 points*

Description

These are the angles used in `STARbar` when `integration = "Turtlesky"`.

Usage

```
data(turtle)
```

Format

A data frame with 59 observations on the following 2 variables.

`altitude` a numeric vector

`azimuth` a numeric vector

`turtle244` *A turtle sky with 244 points*

Description

Not currently used.

Usage

```
data(turtle244)
```

Format

A data frame with 244 observations on the following 2 variables.

`altitude` a numeric vector

`azimuth` a numeric vector

---

turtle482 *A turtle sky with 482 points*

Description

A set of points that are uniformly distributed across the hemisphere, so that each point represents (approx.) the same solid angle. Used in the diffuse radiation calculations in `YplantDay`.

Usage

```
data(turtle482)
```

Format

A data frame with 482 observations on the following 2 variables.

`altitude` a numeric vector

`azimuth` a numeric vector

---

viewplot *Make a three panel plot of a 3D plant*

Description

Three plots of a 3D plant: views from the east, south and from above. This is a lame way to plot the plant, as the stems are always plotted on top (whether or not they are visible). It is available for quick plotting, and for `makereport`, as it does not require the `rgl` package.

See `plot.plant3d` for more advanced, high quality, plotting of plants.

Usage

```
viewplot(plant, side=c("east","south","above"), stems=TRUE, autopar=TRUE)
```

Arguments

`plant`	An object of class 'plant3d' (see `constructplant`).
`side`	Which side to plot (can specify more than 1).
`stems`	If TRUE, plots the stem sections (always on top, lame).
`autopar`	If TRUE, tries to guess how to split up the plotting device.

Details

This function plots the plant from above, east and west views. Stems are also plotted, as opposed to the standard plot of a projected plant (see projectplant).

Note

This function is called by makereport.

Author(s)

Remko Duursma

See Also

plot.plant3d,projectplant

Examples

```
# Toona australis from above
viewplot(toona, "above")
```

---

xmastime *Is it xmas yet?*

Description

Prettify your plant.

Usage

```
xmastime(plant,
palette = c("red", "blue2", "yellow", "darkorchid3", "gold1"),
zadj = 1.5,
nballsfrac = 0.2,
ballradiusrel = 0.25)
```

Arguments

plant	A 'plant3d' object (see constructplant).
palette	Colors.
zadj	Adjust relative height to leaf height.
nballsfrac	Fraction of leaves with ornaments
ballradiusrel	Relative radius of ornaments

Author(s)

Santa

`yplantaltaz` *Altitude and azimuth angles*

Description

The 160 angles used in the original Yplant (Pearcy & Yang 1996), in radians. Used in `STARbar` when `integration = "Yplant"`.

Format

A data frame with 160 observations on the following 2 variables.

`azimuth` Azimuth angle (radians)

`altitude` Altitude (radians)

`YplantDay` *Run a simulation over a day with YplantQMC*

Description

Interface to daily simulations with YplantQMC. Two objects are required to run the simulation: a `plant3d` object, containing the plant structure information, and a `met` object, containing weather data, solar position, and number of timesteps.

Optionally, a `phy` object is used which contains the leaf gas exchange model for the simulation, to calculate photosynthesis (and possibly transpiration rate) from light capture and other weather variables.

Also optional is the use of a `hemi` object, which specifies shading by a canopy.

If you don't know where to start, run the example at the bottom of this page.

Usage

```
## S3 method for class 'plant3d'
YplantDay(x, met, phy = NULL, hemi = NULL,quiet=FALSE,
writePSR=TRUE, writeOUT=FALSE, ...)
## S3 method for class 'plant3dlist'
YplantDay(x, met, phy=NULL, hemi=NULL, ...)
```

Arguments

x	An object of class 'plant3d' or 'plant3dlist' (see constructplant and readplantlist).
met	An object of class 'ypmet', see setMet
phy	An object of class 'ypphy', see setPhy
hemi	An object of class 'yphemi', see setHemi
quiet	If TRUE, does not write messages to the console.
writePSR	If TRUE, writes a PSR output file.
writeOUT	If TRUE, writes an OUT output file.
...	Further arguments passed to runYplant

Details

See the arguments list above for the functions that are used to generate each of the four objects. Note that the plant and met objects are required, and phy and hemi are optional.

This function is a user-friendly wrapper for runYplant. That function should be used for all advanced simulations.

Value

The YplantDay functions returns a list of class yplantsim, which has print and plot methods (see Examples).

The list has the following components:

plant The plant object used in the simulation

phy If provided, the phy object used in the simulation

hemi If provided, the hemi object used in the simulation

outdata A very lengthy dataframe with all results (see below)

nsteps Number of timesteps

psrdata Totals and averages by timestep (dataframe), see psrdata

met The met object used in the simulation

The outdata dataframe in the yplantsim object lists results for individual leaves, has the following variables.

timeofday Time of day for current timestep (hours)

leafnr Leaf number

timestep Length of current timestep (seconds)

PAR0 Above-canopy PAR

PARleaf Total PAR absorption

PARdir Direct solar radiation PAR absorption

PARdiff Diffuse PAR absorption

reldiff Relative diffuse radiation absorption (0-1).

`reldir` Relative direct radiation absorption (0-1).

`LA` Individual leaf area (mm2)

`LAproj` Projected leaf area (mm2)

`LAsunlit` Sunlit, or 'displayed' leaf area (mm2)

`A` CO2 assimilation rate (mu mol m-2 s-1)

`E` Transpiration rate (mmol m-2 s-1)

`gs` Stomatal conductance (mol m-2 s-1)

`A0` CO2 assimilation rate for a horizontal unshaded leaf (mu mol m-2 s-1)

Where PAR is photosynthetically active radiation (mu mol m-2 s-1).

The absorptions `reldiff` and `reldir` are relative to an unshaded horizontal surface.

To extract relative diffuse radiation absorption from an `yplantsim` object, for example:

```
mysim <- YplantDay(myplant, mymet)
reldif <- mysim$outdata$reldiff
```

Author(s)

Remko Duursma

See Also

`runYplant`,`makereport`

Examples

```
## Not run:
# Set location,
southernfrance <- setLocation(lat=44)

# A daily weather object, use a constant beam fraction of 0.4.
sunnyday <- setMet(southernfrance, month=6, day=21, nsteps=12, Tmin=9, Tmax=29, PARday=22,
fbeamday=0.4, fbeammethod="constant")

# Light response curve:
toonalrc <- setPhy("lightresponse",
leafpars=list(Amax=14.5, Rd=1.4, phi=0.05, theta=0.5, reflec=0.1, transmit=0.05))

# Run YplantQMC for a day. Use the built-in 'largegap' hemiphoto.
toonarun <- YplantDay(toona, sunnyday, toonalrc, largegap)

## End(Not run)
```

zenaz *Calculates position of the sun*

Description

Calculates the zenith and azimuth angle of the position of the sun, based mostly on routines from Iqbal (1983).

Usage

```
zenaz(year = 2012,
month = 4,
day = 1,
lat = -33.6,
long = 150.7,
tzlong = long,
KHRS = 24,
timeofday = NA,
LAT = FALSE)
```

Arguments

year	YYYY - to account for eccentricity (small effect).
month	Month number
day	Day of month number
lat	Latitude, degrees.
long	Longitude, degrees. Optional (only used for local apparent time correction.)
tzlong	Longitude of the nearest timezone border
KHRS	Number of timesteps in a day (optional). See Details.
timeofday	Optional, time of day (in hours) (a vector of any length) to calculate the position of the sun
LAT	Logical (default=FALSE). Are the times of day given in 'local apparent time'?

Details

By default, it is assumed that the time of day is not given in local apparent time (LAT, also known as 'solar time'). To convert the standard time to LAT, the longitude of the location, and the longitude of the nearest time zone border must be given.

Alternatively, use LAT=TRUE to specify that the time of day is in LAT (that is, solar maximum occurs exactly at noon).

The user can specify a number of timesteps (KHRS), so that the solar positions are calculated for the midpoint of each timestep (this is used within YplantQMC). Alternatively, specify timeofday directly.

Value

A list with the following components:

`hour` Time in decimal hours

`altitude` Solar altitude (degrees)

`azimuth` Solar azimuth (degrees. N=0, E=90)

`daylength` Day length in hours

`sunset` Time of sunset (hours)

`zenrad` Solar zenith position (radians)

Note

This routine is no doubt less accurate that the NOAA routines provided by the `solarpos` function in the `maptools` package. It is easier to use, though.

Author(s)

Remko Duursma, based mostly on original FORTRAN code by Belinda Medlyn.

References

Iqbal, B., 1983. An Introduction to Solar Radiation. Academic Press, New York, 386 pp

See Also

`setHemi`

Examples

```
# Simple use
zenaz(month=8, day=16, timeofday=12, lat=-33)

# Get half-hourly solar positions
hourpos <- zenaz(month=2, day=16, KHRS=48, lat=-33, long=155, tzlong=150)
 with(hourpos, plot(hour, altitude, type='o',ylab=expression(Altitude~(degree))))
```

Index