Package 'BioCro'

August 18, 2014

Version 0.92

Title BioCro Crop and Agroecosystem Simulator

Description Simulation of C4 Crops and Coppice Trees

Author
Fernando Miguez <femiguez@iastate.edu>,Deepak Jaiswal <djaiswal@illinois.edu>,Dan Wang <dwng@

Maintainer David LeBauer <dlebauer@illinois.edu>

Depends R(>= 2.5.0), lattice, data.table

Suggests coda, knitr, roxygen2(<= 3.1.0)

Imports stats, lattice, data.table

VignetteBuilder knitr

License FreeBSD + file LICENSE

R topics documented:

`aci` *A/Ci curves*

Description

Four A/Ci (assimilation vs. intercellular CO2) curves.

Format

A data frame with 32 observations on the following 7 variables.

list('ID') Identification for each curve. a numeric vector

list('Photo') Assimilation. a numeric vector

list('PARi') Incident Photosynthetic Active Radiation. a numeric vector

list('Tleaf') Temperature of the leaf. a numeric vector

list('RH_S') Realtive humidity. a numeric vector

list('Ci') Intercellular CO2. a numeric vector

list('CO2_R') Reference CO2. a numeric vector

Details

Measurements taken on Miscanthus x giganteus.

Source

Measurements taken by Dandan Wang.

References

Dandan Wang

Examples

```
data(aci)
plotAC(aci)
```

`annualDB` *Miscanthus dry biomass data.*

Description

The first column is the thermal time. The second, third, fourth and fifth columns are miscanthus stem, leaf, root and rhizome dry biomass in Mg ha^{-1} (root is missing). The sixth column is the leaf area index. The `annualDB.c` version is altered so that root biomass is not missing and LAI is smaller. The purpose of this last modification is for testing some functions.

Format

data frame of dimensions 5 by 6.

Source

Clive Beale and Stephen Long. (1997) Seasonal dynamics of nutrient accumulation and partitioning in the perennial C4 grasses Miscanthus x giganteus and Spartina cynosuroides. Biomass and Bioenergy. 419-428.

aq *A/Q curves*

Description

Example of A/Q curves which serves as a template for using the `Opc4photo` and `mOpc4photo` functions.

Format

A data frame with 64 observations on the following 6 variables.

list('ID') a numeric vector

list('trt') a factor with levels `mxg` `swg`

list('A') a numeric vector. Assimilation

list('PARi') a numeric vector. Photosynthetic Active Radiation (incident).

list('Tleaf') a numeric vector. Temperature of the leaf.

list('RH_S') a numeric vector. Relative humidity (fraction).

Details

`swg` stand for switchgrass (Panicum virgatum) `mxg` stands for miscanthus (Miscanthus x gignateus)

above ~~

Source

Data based on measurements made by Dandan Wang. reference to a publication or URL from which the data were obtained ~~

References

Dandan Wang

Examples

```
data(aq)
plotAQ(aq)
```

`ardBuck` *Arden Buck Equation from Buck Research Manual (1996) http://cires.colorado.edu/~voemel/vp.html*

Description

Arden Buck Equation from Buck Research Manual (1996) http://cires.colorado.edu/~voemel/vp.html

Usage

```
ardBuck(Tcelsius)
```

Arguments

`Tcelsius`	the temperature in degrees C

`BioGro` *Biomass crops growth simulation*

Description

Simulates dry biomass growth during an entire growing season. It represents an integration of the photosynthesis function `c4photo`, canopy evapo/transpiration `CanA`, the multilayer canopy model `sunML` and a dry biomass partitioning calendar and senescence. It also considers, carbon and nitrogen cycles and water and nitrogen limitations.

Usage

```
BioGro(WetDat, day1 = NULL, dayn = NULL, timestep = 1, lat = 40,
  iRhizome = 7, irtl = 1e-04, canopyControl = list(),
  seneControl = list(), photoControl = list(), phenoControl = list(),
  soilControl = list(), nitroControl = list(), centuryControl = list())
```

Arguments

`WetDat`	weather data as produced by the `weach` function.
`day1`	first day of the growing season, (1–365).
`dayn`	last day of the growing season, (1–365, but larger than `day1`). See details.
`timestep`	Simulation timestep, the default of 1 requires houlry weather data. A value of 3 would require weather data every 3 hours. This number should be a divisor of 24.
`lat`	latitude, default 40.
`iRhizome`	initial dry biomass of the Rhizome (Mg ha^{-1}).

`irtl` Initial rhizome proportion that becomes leaf. This should not typically be changed, but it can be used to indirectly control the effect of planting density.

`canopyControl`
List that controls aspects of the canopy simulation. It should be supplied through the `canopyParms` function.

`Sp` (specific leaf area) here the units are ha Mg^{-1}. If you have data in m^2 of leaf per kg of dry matter (e.g. 15) then divide by 10 before inputting this coefficient.

`nlayers` (number of layers of the canopy) Maximum 50. To increase the number of layers (more than 50) the `C` source code needs to be changed slightly.

`kd` (extinction coefficient for diffuse light) between 0 and 1.

`mResp` (maintenance respiration) a vector of length 2 with the first component for leaf and stem and the second component for rhizome and root.

`seneControl` List that controls aspects of senescence simulation. It should be supplied through the `seneParms` function.

`senLeaf` Thermal time at which leaf senescence will start.

`senStem` Thermal time at which stem senescence will start.

`senRoot` Thermal time at which root senescence will start.

`senRhizome` Thermal time at which rhizome senescence will start.

`photoControl` List that controls aspects of photosynthesis simulation. It should be supplied through the `photoParms` function.

`vmax` Vmax passed to the `c4photo` function.

`alpha` alpha parameter passed to the `c4photo` function.

`kparm` kparm parameter passed to the `c4photo` function.

`theta` theta parameter passed to the `c4photo` function.

`beta` beta parameter passed to the `c4photo` function.

`Rd` Rd parameter passed to the `c4photo` function.

`Catm` Catm parameter passed to the `c4photo` function.

`b0` b0 parameter passed to the `c4photo` function.

`b1` b1 parameter passed to the `c4photo` function.

`phenoControl` List that controls aspects of the crop phenology. It should be supplied through the `phenoParms` function.

`tp1-tp6` thermal times which determine the time elapsed between phenological stages. Between 0 and tp1 is the juvenile stage. etc.

`kLeaf1-6` proportion of the carbon that is allocated to leaf for phenological stages 1 through 6.

`kStem1-6` proportion of the carbon that is allocated to stem for phenological stages 1 through 6.

`kRoot1-6` proportion of the carbon that is allocated to root for phenological stages 1 through 6.

`kRhizome1-6` proportion of the carbon that is allocated to rhizome for phenological stages 1 through 6.

`kGrain1-6` proportion of the carbon that is allocated to grain for phenological stages 1 through 6. At the moment only the last stage (i.e. 6 or post-flowering) is allowed to be larger than zero. An error will be returned if kGrain1-5 are different from zero.

`soilControl` List that controls aspects of the soil environment. It should be supplied through the `soilParms` function.

`FieldC` Field capacity. This can be used to override the defaults possible from the soil types (see `showSoilType`).

`WiltP` Wilting point. This can be used to override the defaults possible from the soil types (see `showSoilType`).

`phi1` Parameter which controls the spread of the logistic function. See `wtrstr` for more details.

`phi2` Parameter which controls the reduction of the leaf area growth due to water stress. See `wtrstr` for more details.

`soilDepth` Maximum depth of the soil that the roots have access to (i.e. rooting depth).

`iWatCont` Initial water content of the soil the first day of the growing season. It can be a single value or a vector for the number of layers specified.

`soilType` Soil type, default is 6 (a more typical soil would be 3). To see details use the function `showSoilType`.

`soilLayer` Integer between 1 and 50. The default is 5. If only one soil layer is used the behavior can be quite different.

`soilDepths` Intervals for the soil layers.

`wsFun` one of 'logistic','linear','exp' or 'none'. Controls the method for the relationship between soil water content and water stress factor.

`scsf` stomatal conductance sensitivity factor (default = 1). This is an empirical coefficient that needs to be adjusted for different species.

`rfl` Root factor lambda. A Poisson distribution is used to simulate the distribution of roots in the soil profile and this parameter can be used to change the lambda parameter of the Poisson.

`rsec` Radiation soil evaporation coefficient. Empirical coefficient used in the incidence of direct radiation on soil evaporation.

`rsdf` Root soil depth factor. Empirical coefficient used in calculating the depth of roots as a function of root biomass.

`nitroControl` List that controls aspects of the nitrogen environment. It should be supplied through the `nitrolParms` function.

`iLeafN` initial value of leaf nitrogen (g m-2).

`kLN` coefficient of decrease in leaf nitrogen during the growing season. The equation is LN = iLeafN * (Stem + Leaf)^-kLN .

`Vmax.b1` slope which determines the effect of leaf nitrogen on Vmax.

`alpha.b1` slope which controls the effect of leaf nitrogen on alpha.

`centuryControl`

List that controls aspects of the Century model for carbon and nitrogen dynamics in the soil. It should be supplied through the `centuryParms` function.

`SC1-9` Soil carbon pools in the soil. SC1: Structural surface litter. SC2: Metabolic surface litter. SC3: Structural root litter. SC4: Metabolic root litter. SC5: Surface microbe. SC6: Soil microbe. SC7: Slow carbon. SC8: Passive carbon. SC9: Leached carbon.

`LeafL.Ln` Leaf litter lignin content.

`StemL.Ln` Stem litter lignin content.
`RootL.Ln` Root litter lignin content.
`RhizomeL.Ln` Rhizome litter lignin content.
`LeafL.N` Leaf litter nitrogen content.
`StemL.N` Stem litter nitrogen content.
`RootL.N` Root litter nitrogen content.
`RhizomeL.N` Rhizome litter nitrogen content.
`Nfert` Nitrogen from a fertilizer source.
`iMinN` Initial value for the mineral nitrogen pool.
`Litter` Initial values of litter (leaf, stem, root, rhizome).
`timestep` currently either week (default) or day.

Value

a `list` structure with components

Examples

```
## Not run:
data(weather05)

res0 <- BioGro(weather05)

plot(res0)

## Looking at the soil model

res1 <- BioGro(weather05, soilControl = soilParms(soilLayers = 6))
plot(res1, plot.kind='SW') ## Without hydraulic distribution
res2 <- BioGro(weather05, soilControl = soilParms(soilLayers = 6, hydrDist=TRUE))
plot(res2, plot.kind='SW') ## With hydraulic distribution


## Example of user defined soil parameters.
## The effect of phi2 on yield and soil water content

ll.0 <- soilParms(FieldC=0.37,WiltP=0.2,phi2=1)
ll.1 <- soilParms(FieldC=0.37,WiltP=0.2,phi2=2)
ll.2 <- soilParms(FieldC=0.37,WiltP=0.2,phi2=3)
ll.3 <- soilParms(FieldC=0.37,WiltP=0.2,phi2=4)

ans.0 <- BioGro(weather05,soilControl=ll.0)
ans.1 <- BioGro(weather05,soilControl=ll.1)
ans.2 <- BioGro(weather05,soilControl=ll.2)
ans.3 <-BioGro(weather05,soilControl=ll.3)

xyplot(ans.0$SoilWatCont +
       ans.1$SoilWatCont +
       ans.2$SoilWatCont +
       ans.3$SoilWatCont ~ ans.0$DayofYear,
```

```
        type='l',
        ylab='Soil water Content (fraction)',
        xlab='DOY')

## Compare LAI

xyplot(ans.0$LAI +
        ans.1$LAI +
        ans.2$LAI +
        ans.3$LAI ~ ans.0$DayofYear,
        type='l',
        ylab='Leaf Area Index',
        xlab='DOY')



## End(Not run)
```

`c3CanA` *Simulates canopy assimilation for C3 canopies*

Description

It represents an integration of the photosynthesis function `c3photo`, canopy evapo/transpiration and the multilayer canopy model `sunML`.

Usage

```
c3CanA(lai, doy, hr, solar, temp, rh, windspeed, lat = 40, nlayers = 8,
  kd = 0.1, heightFactor = 3, c3photoControl = list(),
  lnControl = list(), StomWS = 1)
```

Arguments

`lai`	leaf area index.
`doy`	day of the year, (1–365).
`hr`	hour of the day, (0–23).
`solar`	solar radiation (μ mol m^{-2} s^{-1}).
`temp`	temperature (Celsius).
`rh`	relative humidity (0–1).
`windspeed`	wind speed (m s^{-1}).
`lat`	latitude.
`nlayers`	number of layers in the simulation of the canopy (max allowed is 50).
`kd`	Ligth extinction coefficient for diffuse light.

`heightFactor` Height Factor. Divide LAI by this number to get the height of a crop.

`c3photoControl`
list that sets the photosynthesis parameters for c3 plants through the c3photoParms function

`lnControl` list that sets the leaf nitrogen parameters.

LeafN: Initial value of leaf nitrogen (g m-2).

kpLN: coefficient of decrease in leaf nitrogen during the growing season. The equation is LN = iLeafN * exp(-kLN * TTc).

lnFun: controls whether there is a decline in leaf nitrogen with the depth of the canopy. 'none' means no decline, 'linear' means a linear decline controlled by the following two parameters.

lnb0: Intercept of the linear decline of leaf nitrogen in the depth of the canopy.

lnb1: Slope of the linear decline of leaf nitrogen in the depth of the canopy. The equation is 'vmax = leafN_lay * lnb1 + lnb0'.

Details

The photosynthesis function is modeled after the version in WIMOVAC. This is based on the well known Farquar model.

Value

`list`

returns a list with several elements

CanopyAssim: hourly canopy assimilation ($Mgha^{-1}\ hr^{-1}$)

CanopyTrans: hourly canopy transpiration ($Mgha^{-1}\ hr^{-1}$)

CanopyCond: hourly canopy conductance (units ?)

TranEpen: hourly canopy transpiration according to Penman ($Mgha^{-1}\ hr^{-1}$)

TranEpen: hourly canopy transpiration according to Priestly ($Mgha^{-1}\ hr^{-1}$)

LayMat: hourly by Layer matrix containing details of the calculations by layer (each layer is a row). col1: Direct Irradiance col2: Diffuse Irradiance col3: Leaf area in the sun col4: Leaf area in the shade col5: Transpiration of leaf area in the sun col6: Transpiration of leaf area in the shade col7: Assimilation of leaf area in the sun col8: Assimilation of leaf area in the shade col9: Difference in temperature between the leaf and the air (i.e. TLeaf - TAir) for leaves in sun. col10: Difference in temperature between the leaf and the air (i.e. TLeaf - TAir) for leaves in shade. col11: Stomatal conductance for leaves in the sun col12: Stomatal conductance for leaves in the shade col13: Nitrogen concentration in the leaf (g m^-2) col14: Vmax value as depending on leaf nitrogen

Author(s)

Fernando E. Miguez

References

Farquhar model site here ~

Examples

```
data(doy124)
tmp <- numeric(24)

for(i in 1:24){
   lai <- doy124[i,1]
   doy <- doy124[i,3]
   hr  <- doy124[i,4]
 solar <- doy124[i,5]
  temp <- doy124[i,6]
    rh <- doy124[i,7]
    ws <- doy124[i,8]

  tmp[i] <- c3CanA(lai,doy,hr,solar,temp,rh,ws)$CanopyAssim

}

plot(c(0:23),tmp,
            type='l',lwd=2,
            xlab='Hour',
            ylab=expression(paste('Canopy assimilation (Mg  ',
            ha^-2,' ',h^-1,')')))
```

`c3photo` *Simulates C3 photosynthesis*

Description

Simulates coupled assimilation and stomatal conductance based on Farquhar and Ball-Berry.

Usage

```
c3photo(Qp, Tl, RH, vcmax = 100, jmax = 180, Rd = 1.1, Catm = 380,
  O2 = 210, b0 = 0.08, b1 = 5, theta = 0.7, StomWS = 1, ws = c("gs",
  "vmax"))
```

Arguments

`Qp`	Quantum flux
`Tl`	Leaf temperature
`RH`	Relative humidity (fraction – 0-1)
`vcmax`	Maximum rate of carboxylation
`jmax`	Maximum rate of electron transport
`Rd`	Leaf dark respiration
`Catm`	Atmospheric carbon dioxide.
`O2`	Atmospheric Oxygen concentration (mmol/mol)

b0	Intercept for the Ball-Berry model
b1	Slope for the Ball-Berry model
theta	Curvature parameter

Value

A list

Note

~~further notes~~ ## Additional notes about assumptions

Author(s)

Fernando E. Miguez

References

Farquhar (1980) Ball-Berry (1987)

See Also

See Also Opc3photo

Examples

```
## Testing the c3photo function
## First example: looking at the effect of changing Vcmax
Qps <- seq(0,2000,10)
Tls <- seq(0,55,5)
rhs <- c(0.7)
dat1 <- data.frame(expand.grid(Qp=Qps,Tl=Tls,RH=rhs))

res1 <- c3photo(dat1$Qp,dat1$Tl,dat1$RH) ## default Vcmax = 100
res2 <- c3photo(dat1$Qp,dat1$Tl,dat1$RH,vcmax=120)

## Plot comparing alpha 0.04 vs. 0.06 for a range of conditions
xyplot(res1$Assim + res2$Assim ~ Qp | factor(Tl) , data = dat1,
            type='l',col=c('blue','green'),lwd=2,
            ylab=expression(paste('Assimilation (',
                 mu,mol,' ',m^-2,' ',s^-1,')')),
             xlab=expression(paste('Quantum flux (',
                 mu,mol,' ',m^-2,' ',s^-1,')')),
            key=list(text=list(c('Vcmax 100','Vcmax 120')),
              lines=TRUE,col=c('blue','green'),lwd=2))

## Second example: looking at the effect of changing Jmax
## Plot comparing Jmax 300 vs. 100 for a range of conditions

res1 <- c3photo(dat1$Qp,dat1$Tl,dat1$RH) ## default Jmax = 300
res2 <- c3photo(dat1$Qp,dat1$Tl,dat1$RH,jmax=100)
```

```
xyplot(res1$Assim + res2$Assim ~ Qp | factor(Tl) , data = dat1,
           type='l',col=c('blue','green'),lwd=2,
            ylab=expression(paste('Assimilation (',
                 mu,mol,' ',m^-2,' ',s^-1,')')),
             xlab=expression(paste('Quantum flux (',
                 mu,mol,' ',m^-2,' ',s^-1,')')),
            key=list(text=list(c('Jmax 300','Jmax 100')),
              lines=TRUE,col=c('blue','green'),lwd=2))

## A/Ci curve

Ca <- seq(20,1000,length.out=50)
dat2 <- data.frame(Qp=rep(700,50), Tl=rep(25,50), rh=rep(0.7,50))
res1 <- c3photo(dat2$Qp, dat2$Tl, dat2$rh, Catm = Ca)
res2 <- c3photo(dat2$Qp, dat2$Tl, dat2$rh, Catm = Ca, vcmax = 70)

xyplot(res1$Assim ~ res1$Ci,
           lwd=2,
           panel = function(x,y,...){
                   panel.xyplot(x,y,type='l',col='blue',...)
                   panel.xyplot(res2$Ci,res2$Assim, type='l', col =
           'green',...)
           },
            ylab=expression(paste('Assimilation (',
                 mu,mol,' ',m^-2,' ',s^-1,')')))
```

`c4photo` *Coupled photosynthesis-stomatal conductance simulation*

Description

The mathematical model is based on Collatz et al (1992) (see References). Stomatal conductance is based on code provided by Joe Berry.

Usage

```
c4photo(Qp, Tl, RH, vmax = 39, alpha = 0.04, kparm = 0.7, theta = 0.83,
  beta = 0.93, Rd = 0.8, UPPERTEMP = 37.5, LOWERTEMP = 3, Catm = 380,
  b0 = 0.08, b1 = 3, StomWS = 1, ws = c("gs", "vmax"))
```

Arguments

`Qp`	quantum flux (direct light), (μ mol m^{-2} s^{-1}).
`Tl`	temperature of the leaf (Celsius).
`RH`	relative humidity (as a fraction, i.e. 0-1).
`vmax`	maximum carboxylation of Rubisco according to the Collatz model.

alpha	alpha parameter according to the Collatz model. Initial slope of the response to Irradiance.
kparm	k parameter according to the Collatz model. Initial slope of the response to CO2.
theta	theta parameter according to the Collatz model. Curvature for light response.
beta	beta parameter according to the Collatz model. Curvature for response to CO2.
Rd	Rd parameter according to the Collatz model. Dark respiration.
UPPERTEMP	
LOWERTEMP	
Catm	Atmospheric CO2 in ppm (or μmol/mol).
b0	intercept for the Ball-Berry stomatal conductance model.
b1	slope for the Ball-Berry stomatal conductance model.
StomWS	coefficient which controls the effect of water stress on stomatal conductance and assimilation.
ws	option to control whether the water stress factor is applied to stomatal conductance ('gs') or to Vmax ('vmax').

Value

a list structure with components

References

G. Collatz, M. Ribas-Carbo, J. Berry. (1992). Coupled photosynthesis-stomatal conductance model for leaves of C4 plants. *Australian Journal of Plant Physiology* 519–538.

See Also

eC4photo

Examples

```
## Not run:
     ## First example: looking at the effect of changing alpha
      Qps <- seq(0,2000,10)
      Tls <- seq(0,55,5)
      rhs <- c(0.7)
      dat1 <- data.frame(expand.grid(Qp=Qps,Tl=Tls,RH=rhs))
      res1 <- c4photo(dat1$Qp,dat1$Tl,dat1$RH) ## default alpha = 0.04
      res2 <- c4photo(dat1$Qp,dat1$Tl,dat1$RH,alpha=0.06)

     ## Plot comparing alpha 0.04 vs. 0.06 for a range of conditions
     xyplot(res1$Assim + res2$Assim ~ Qp | factor(Tl) , data = dat1,
            type='l',col=c('blue','green'),lwd=2,
            ylab=expression(paste('Assimilation (',
                 mu,mol,' ',m^-2,' ',s^-1,')')),
             xlab=expression(paste('Quantum flux (',
                 mu,mol,' ',m^-2,' ',s^-1,')')),
```

```
       key=list(text=list(c('alpha 0.04','alpha 0.06')),
         lines=TRUE,col=c('blue','green'),lwd=2))

 ## Second example: looking at the effect of changing vmax
 ## Plot comparing Vmax 39 vs. 50 for a range of conditions

  res1 <- c4photo(dat1$Qp,dat1$Tl,dat1$RH) ## default Vmax = 39
  res2 <- c4photo(dat1$Qp,dat1$Tl,dat1$RH,vmax=50)

 xyplot(res1$Assim + res2$Assim ~ Qp | factor(Tl) , data = dat1,
       type='l',col=c('blue','green'),lwd=2,
       ylab=expression(paste('Assimilation (',
            mu,mol,' ',m^-2,' ',s^-1,')')),
        xlab=expression(paste('Quantum flux (',
            mu,mol,' ',m^-2,' ',s^-1,')')),
       key=list(text=list(c('Vmax 39','Vmax 50')),
         lines=TRUE,col=c('blue','green'),lwd=2))

 ## Small effect of low RH on  Assim
  Qps <- seq(0,2000,10)
  Tls <- seq(0,55,5)
  rhs <- c(0.2,0.9)
  dat1 <- data.frame(expand.grid(Qp=Qps,Tl=Tls,RH=rhs))
  res1 <- c4photo(dat1$Qp,dat1$Tl,dat1$RH)
 # plot for Assimilation and two RH
  xyplot(res1$Assim ~ Qp | factor(Tl) , data = dat1,
         groups=RH, type='l',
         col=c('blue','green'),lwd=2,
         ylab=expression(paste('Assimilation (',
             mu,mol,' ',m^-2,' ',s^-1,')')),
         xlab=expression(paste('Quantum flux (',
             mu,mol,' ',m^-2,' ',s^-1,')')),
         key=list(text=list(c('RH 20%','RH 90%')),
                    lines=TRUE,col=c('blue','green'),
                    lwd=2))

 ## Effect of the previous runs on Stomatal conductance

 xyplot(res1$Gs ~ Qp | factor(Tl) , data = dat1,
        type='l', groups=RH,
        col=c('blue','green'),lwd=2,
        ylab=expression(paste('Stomatal Conductance (',
                      mu,mol,' ',m^-2,' ',s^-1,')')),
        xlab=expression(paste('Quantum flux (',
                      mu,mol,' ',m^-2,' ',s^-1,')')),
        key=list(text=list(c('RH 20%','RH 90%')),
                  lines=TRUE,col=c('blue','green'),
                  lwd=2))


## A Ci curve for the Collatz model
## Assuming constant values of Qp, Temp, and RH
## Notice the effect of the different kparm
```

```
## The loop is needed because the length of Ca
## should be the same as Qp

Ca <- seq(15,400,5)

res1 <- numeric(length(Ca))
res2 <- numeric(length(Ca))
for(i in 1:length(Ca)){
  res1[i] <- c4photo(1500,25,0.7,Catm=Ca[i])$Assim
  res2[i] <- c4photo(1500,25,0.7,Catm=Ca[i],kparm=0.8)$Assim
}

xyplot(res1 + res2 ~ Ca ,type='l',lwd=2,
       col=c('blue','green'),
     xlab=expression(paste(CO[2],' (ppm)')),
     ylab=expression(paste('Assimilation (',
         mu,mol,' ',m^-2,' ',s^-1,')')),
     key=list(text=list(c('kparm 0.7','kparm 0.8')),
                       lines=TRUE,col=c('blue','green'),
                       lwd=2))

## Effect of Reduction in Assimilation due to
## water stress

Qps <- seq(0,2000,10)
Tls <- seq(0,55,5)
rhs <- c(0.7)
dat1 <- data.frame(expand.grid(Qp=Qps,Tl=Tls,RH=rhs))
res1 <- c4photo(dat1$Qp,dat1$Tl,dat1$RH) ## default StomWS = 1 No stress
res2 <- c4photo(dat1$Qp,dat1$Tl,dat1$RH,StomWS=0.5)

## Plot comparing StomWS = 1 vs. 0.5 for a range of conditions
xyplot(res1$Assim + res2$Assim ~ Qp | factor(Tl) , data = dat1,
       type='l',col=c('blue','green'),lwd=2,
       ylab=expression(paste('Assimilation (',
           mu,mol,' ',m^-2,' ',s^-1,')')),
       xlab=expression(paste('Quantum flux (',
            mu,mol,' ',m^-2,' ',s^-1,')')),
       key=list(text=list(c('StomWS 1','StomWS 0.5')),
           lines=TRUE,col=c('blue','green'),lwd=2))


## Effect on Stomatal Conductance
## Plot comparing StomWS = 1 vs. 0.5 for a range of conditions
xyplot(res1$Gs + res2$Gs ~ Qp | factor(Tl) , data = dat1,
        type='l',col=c('blue','green'),lwd=2,
        ylab=expression(paste('Stomatal Conductance (mmol ',
          m^-2,' ',s^-1,')')),
        xlab=expression(paste('Quantum flux (',
          mu,mol,' ',m^-2,' ',s^-1,')')),
        key=list(text=list(c('StomWS 1','StomWS 0.5')),
           lines=TRUE,col=c('blue','green'),lwd=2))
```

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

CanA *Simulates canopy assimilation*

Description

It represents an integration of the photosynthesis function `c4photo`, canopy evapo/transpiration and the multilayer canopy model `sunML`.

Usage

```
CanA(lai, doy, hr, solar, temp, rh, windspeed, lat = 40, nlayers = 8,
  kd = 0.1, StomataWS = 1, chi.l = 1, heightFactor = 3,
  photoControl = list(), lnControl = list(), units = c("kg/m2/hr",
  "Mg/ha/hr"))
```

Arguments

`lai`	leaf area index.
`doy`	day of the year, (1–365).
`hr`	hour of the day, (0–23).
`solar`	solar radiation (μ mol m^{-2} s^{-1}).
`temp`	temperature (Celsius).
`rh`	relative humidity (0–1).
`windspeed`	wind speed (m s^{-1}).
`lat`	latitude.
`nlayers`	number of layers in the simulation of the canopy (max allowed is 50).
`kd`	Ligth extinction coefficient for diffuse light.
`StomataWS`	coefficient controlling the effect of water stress on stomatal conductance and assimilation.
`heightFactor`	Height Factor. Divide LAI by this number to get the height of a crop.
`photoControl`	list that sets the photosynthesis parameters. See `BioGro`.
`lnControl`	list that sets the leaf nitrogen parameters. LeafN: Initial value of leaf nitrogen (g m-2). kpLN: coefficient of decrease in leaf nitrogen during the growing season. The equation is LN = iLeafN * exp(-kLN * TTc). lnFun: controls whether there is a decline in leaf nitrogen with the depth of the canopy. 'none' means no decline, 'linear' means a linear decline controlled by the following two parameters. lnb0: Intercept of the linear decline of leaf nitrogen in the depth of the canopy. lnb1: Slope of the linear decline of leaf nitrogen in the depth of the canopy. The equation is 'vmax = leafN_lay * lnb1 + lnb0'.
`units`	Whether to return units in kg/m2/hr or Mg/ha/hr. This is typically run at hourly intervals, that is why the hr is kept, but it could be used with data at finer timesteps and then convert the results.

Value

`list`

returns a list with several elements

CanopyAssim: hourly canopy assimilation ($kgm^{-2}\ hr^{-1}$) or canopy assimilation ($Mgha^{-1}\ hr^{-1}$)

CanopyTrans: hourly canopy transpiration ($kgm^{-2}\ hr^{-1}$) or canopy transpiration ($Mgha^{-1}\ hr^{-1}$)

CanopyCond: hourly canopy conductance (units ?)

TranEpen: hourly canopy transpiration according to Penman ($kgm^{-2}\ hr^{-1}$) or canopy transpiration according to Penman ($Mgha^{-1}\ hr^{-1}$)

TranEpen: hourly canopy transpiration according to Priestly ($kgm^{-2}\ hr^{-1}$) canopy transpiration according to Priestly ($Mgha^{-1}\ hr^{-1}$)

LayMat: hourly by Layer matrix containing details of the calculations by layer (each layer is a row). col1: Direct Irradiance col2: Diffuse Irradiance col3: Leaf area in the sun col4: Leaf area in the shade col5: Transpiration of leaf area in the sun col6: Transpiration of leaf area in the shade col7: Assimilation of leaf area in the sun col8: Assimilation of leaf area in the shade col9: Difference in temperature between the leaf and the air (i.e. TLeaf - TAir) for leaves in sun. col10: Difference in temperature between the leaf and the air (i.e. TLeaf - TAir) for leaves in shade. col11: Stomatal conductance for leaves in the sun col12: Stomatal conductance for leaves in the shade col13: Nitrogen concentration in the leaf (g m^-2) col14: Vmax value as depending on leaf nitrogen

Examples

```
## Not run:
data(doy124)
tmp <- numeric(24)

for(i in 1:24){
   lai <- doy124[i,1]
   doy <- doy124[i,3]
   hr  <- doy124[i,4]
 solar <- doy124[i,5]
  temp <- doy124[i,6]
    rh <- doy124[i,7]
    ws <- doy124[i,8]

  tmp[i] <- CanA(lai,doy,hr,solar,temp,rh,ws)$CanopyAssim

}

plot(c(0:23),tmp,
            type='l',lwd=2,
            xlab='Hour',
            ylab=expression(paste('Canopy assimilation (kg  ',
            m^-2,' ',h^-1,')')))


## End(Not run)
```

caneGro *Simulation of cane, Growth, LAI, Photosynthesis and phenology*

Description

It takes weather data as input (hourly timesteps) and several parameters and it produces phenology, photosynthesis, LAI, etc.

Usage

```
caneGro(WetDat, day1 = NULL, dayn = NULL, timestep = 1, lat = 40,
  iRhizome = 7, irtl = 1e-04, canopyControl = list(),
  seneControl = list(), photoControl = list(), phenoControl = list(),
  soilControl = list(), nitroControl = list(), canephenoControl = list(),
  centuryControl = list(), managementControl = list(),
  frostControl = list())
```

Arguments

WetDat	weather data as produced by the weach function.
plant.day	Planting date (format 0-365)
emerge.day	Emergence date (format 0-365)
harvest.day	Harvest date (format 0-365)
plant.density	Planting density (plants per meter squared, default = 7)
timestep	Simulation timestep, the default of 1 requires houlry weather data. A value of 3 would require weather data every 3 hours. This number should be a divisor of 24.
lat	latitude, default 40.
canopyControl	List that controls aspects of the canopy simulation. It should be supplied through the canopyParms function. Sp (specific leaf area) here the units are ha Mg^{-1}. If you have data in m^2 of leaf per kg of dry matter (e.g. 15) then divide by 10 before inputting this coefficient. SpD decrease of specific leaf area. Empirical parameter. Default 0. example value (1.7e-3). nlayers (number of layers of the canopy) Maximum 50. To increase the number of layers (more than 50) the C source code needs to be changed slightly. kd (extinction coefficient for diffuse light) between 0 and 1. mResp (maintenance respiration) a vector of length 2 with the first component for leaf and stem and the second component for rhizome and root.
caneSeneControl	List that controls aspects of senescence simulation. It should be supplied through the caneSeneParms function.

senLeaf Thermal time at which leaf senescence will start.

senStem Thermal time at which stem senescence will start.

senRoot Thermal time at which root senescence will start.

photoControl List that controls aspects of photosynthesis simulation. It should be supplied through the canePhotoParms function.

vmax Vmax passed to the c4photo function.

alpha alpha parameter passed to the c4photo function.

kparm kparm parameter passed to the c4photo function.

theta theta parameter passed to the c4photo function.

beta beta parameter passed to the c4photo function.

Rd Rd parameter passed to the c4photo function.

UPPERTEMP UPPERTEMP parameter passed to the c4photo function.

LOWERTEMP LOWERTEMP parameter passed to the c4photo function.

Catm Catm parameter passed to the c4photo function.

b0 b0 parameter passed to the c4photo function.

b1 b1 parameter passed to the c4photo function.

canePhenoControl

argument used to pass parameters related to phenology characteristics canePhenoControl here~~

soilControl here~~

nitroControl here~~

centuryControl

here~~

Details

The phenology follows the 'Corn Growth and Development' Iowa State Publication. than the description above ~~

Value

It currently returns a list with the following components

DayofYear	Day of the year (0-365)
Hour	Hour of the day (0-23)
TTTc	Accumulated thermal time
PhenoStage	Phenological stage of the crop
CanopyAssim	Hourly canopy assimilation, (Mg ha^-1 ground hr^-1).
CanopyTrans	Hourly canopy transpiration, (Mg ha^-1 ground hr^-1).
LAI	Leaf Area Index

Author(s)

Fernando E Miguez

See Also

BioGro help, ~~~

Century *This function implements the Century model from Parton.*

Description

Calculates flows of soil organic carbon and nitrogen based on the Century model. There are two versions one written in R (Century) and one in C (CenturyC) they should give the same result. The C version only runs at weekly timesteps.

Usage

```
Century(LeafL, StemL, RootL, RhizL, smoist, stemp, precip, leachWater,
  centuryControl = list(), verbose = FALSE)
```

Arguments

LeafL	Leaf litter.
StemL	Stem litter.
RootL	Root litter.
RhizL	Rhizome litter.
smoist	Soil moisture.
stemp	Soil temperature.
precip	Precipitation.
leachWater	Leached water.
centuryControl	See centuryParms.
verbose	Only used in the R version for debugging.
soilType	See showSoilType.

Details

Most of the details can be found in the papers by Parton et al. (1987, 1988, 1993)

Value

A list with,

Author(s)

Fernando E. Miguez

References

~put references to the literature/web site here ~

Examples

```
Century(0,0,0,0,0.3,5,2,2)$Resp
Century(0,0,0,0,0.3,5,2,2)$MinN
```

`CenturyC` *C version of the Century function*

Description

C version of the Century function

Usage

```
CenturyC(LeafL, StemL, RootL, RhizL, smoist, stemp, precip, leachWater,
  centuryControl = list(), soilType = 0)
```

Arguments

`LeafL`	Leaf litter.
`StemL`	Stem litter.
`RootL`	Root litter.
`RhizL`	Rhizome litter.
`smoist`	Soil moisture.
`stemp`	Soil temperature.
`precip`	Precipitation.
`leachWater`	Leached water.
`centuryControl`	See `centuryParms`.
`soilType`	See `showSoilType`.

CheckLeapYear *This fuction checks if year is a leap year. If yes, then returns 1 or 0.*

Description

This fuction checks if year is a leap year. If yes, then returns 1 or 0.

Usage

```
CheckLeapYear(year)
```

Arguments

year	the year in question

cmi0506 *Weather data*

Description

selected weather data corresponding to the Champaign weather station (IL, U.S.). It has two years: 2005 and 2006. Dimensions: 730 by 11. The columns correspond to the input necessary for the weach function.

Format

data frame of dimensions 730 by 11.

Source

http://www.sws.uiuc.edu/warm/

cmiWet *Weather data*

Description

Layer data for evapo/transpiration. Simulated data to show the result of the EvapoTrans function.

Format

data frame of dimensions 384 by 9.

Details

lfClass: leaf class, 'sun' or 'shade'.

layer: layer in the canopy, 1 to 8.

hour: hour of the day, (0–23).

Rad: direct light.

Itot: total radiation.

Temp: air temperature, (Celsius).

RH: relative humidity, (0–1).

WindSpeed: wind speed, (m s^{-1}).

LAI: leaf area index.

Source

simulated

`CropGro` *Biomass crops growth simulation*

Description

Simulates dry biomass growth during an entire growing season. It represents an integration of the photosynthesis function `c4photo`, canopy evapo/transpiration `CanA`, the multilayer canopy model `sunML` and a dry biomass partitioning calendar and senescence. It also considers, carbon and nitrogen cycles and water and nitrogen limitations.

Usage

```
CropGro(WetDat, day1 = NULL, dayn = NULL, timestep = 1, lat = 40,
  iRhizome = 7, irtl = 1e-04, canopyControl = list(),
  seneControl = list(), photoControl = list(), phenoControl = list(),
  soilControl = list(), nitroControl = list(), SOMpoolsControl = list(),
  SOMAssignParmsControl = list(), GetCropCentStateVarParmsControl = list(),
  GetSoilTextureParmsControl = list(),
  GetBioCroToCropcentParmsControl = list(), GetErosionParmsControl = list(),
  GetC13ParmsControl = list(), GetLeachingParmsControl = list(),
  GetSymbNFixationParmsControl = list(), centuryControl = list())
```

Arguments

`WetDat`	weather data as produced by the `weach` function.
`day1`	first day of the growing season, (1–365).
`dayn`	last day of the growing season, (1–365, but larger than `day1`). See details.

`timestep` Simulation timestep, the default of 1 requires houlry weather data. A value of 3 would require weather data every 3 hours. This number should be a divisor of 24.

`lat` latitude, default 40.

`iRhizome` initial dry biomass of the Rhizome (Mg ha^{-1}).

`irtl` Initial rhizome proportion that becomes leaf. This should not typically be changed, but it can be used to indirectly control the effect of planting density.

`canopyControl`

List that controls aspects of the canopy simulation. It should be supplied through the `canopyParms` function.

`Sp` (specific leaf area) here the units are ha Mg^{-1}. If you have data in m^2 of leaf per kg of dry matter (e.g. 15) then divide by 10 before inputting this coefficient.

`nlayers` (number of layers of the canopy) Maximum 50. To increase the number of layers (more than 50) the `C` source code needs to be changed slightly.

`kd` (extinction coefficient for diffuse light) between 0 and 1.

`mResp` (maintenance respiration) a vector of length 2 with the first component for leaf and stem and the second component for rhizome and root.

`seneControl` List that controls aspects of senescence simulation. It should be supplied through the `seneParms` function.

`senLeaf` Thermal time at which leaf senescence will start.

`senStem` Thermal time at which stem senescence will start.

`senRoot` Thermal time at which root senescence will start.

`senRhizome` Thermal time at which rhizome senescence will start.

`photoControl` List that controls aspects of photosynthesis simulation. It should be supplied through the `photoParms` function.

`vmax` Vmax passed to the `c4photo` function.

`alpha` alpha parameter passed to the `c4photo` function.

`kparm` kparm parameter passed to the `c4photo` function.

`theta` theta parameter passed to the `c4photo` function.

`beta` beta parameter passed to the `c4photo` function.

`Rd` Rd parameter passed to the `c4photo` function.

`Catm` Catm parameter passed to the `c4photo` function.

`b0` b0 parameter passed to the `c4photo` function.

`b1` b1 parameter passed to the `c4photo` function.

`phenoControl` List that controls aspects of the crop phenology. It should be supplied through the `phenoParms` function.

`tp1-tp6` thermal times which determine the time elapsed between phenological stages. Between 0 and tp1 is the juvenile stage. etc.

`kLeaf1-6` proportion of the carbon that is allocated to leaf for phenological stages 1 through 6.

`kStem1-6` proportion of the carbon that is allocated to stem for phenological stages 1 through 6.

`kRoot1-6` proportion of the carbon that is allocated to root for phenological stages 1 through 6.

kRhizome1-6 proportion of the carbon that is allocated to rhizome for phenological stages 1 through 6.

kGrain1-6 proportion of the carbon that is allocated to grain for phenological stages 1 through 6. At the moment only the last stage (i.e. 6 or post-flowering) is allowed to be larger than zero. An error will be returned if kGrain1-5 are different from zero.

soilControl List that controls aspects of the soil environment. It should be supplied through the soilParms function.

FieldC Field capacity. This can be used to override the defaults possible from the soil types (see showSoilType).

WiltP Wilting point. This can be used to override the defaults possible from the soil types (see showSoilType).

phi1 Parameter which controls the spread of the logistic function. See wtrstr for more details.

phi2 Parameter which controls the reduction of the leaf area growth due to water stress. See wtrstr for more details.

soilDepth Maximum depth of the soil that the roots have access to (i.e. rooting depth).

iWatCont Initial water content of the soil the first day of the growing season. It can be a single value or a vector for the number of layers specified.

soilType Soil type, default is 6 (a more typical soil would be 3). To see details use the function showSoilType.

soilLayer Integer between 1 and 50. The default is 5. If only one soil layer is used the behavior can be quite different.

soilDepths Intervals for the soil layers.

wsFun one of 'logistic','linear','exp' or 'none'. Controls the method for the relationship between soil water content and water stress factor.

scsf stomatal conductance sensitivity factor (default = 1). This is an empirical coefficient that needs to be adjusted for different species.

rfl Root factor lambda. A Poisson distribution is used to simulate the distribution of roots in the soil profile and this parameter can be used to change the lambda parameter of the Poisson.

rsec Radiation soil evaporation coefficient. Empirical coefficient used in the incidence of direct radiation on soil evaporation.

rsdf Root soil depth factor. Empirical coefficient used in calculating the depth of roots as a function of root biomass.

nitroControl List that controls aspects of the nitrogen environment. It should be supplied through the nitrolParms function.

iLeafN initial value of leaf nitrogen (g m-2).

kLN coefficient of decrease in leaf nitrogen during the growing season. The equation is LN = iLeafN * (Stem + Leaf)^-kLN .

Vmax.b1 slope which determines the effect of leaf nitrogen on Vmax.

alpha.b1 slope which controls the effect of leaf nitrogen on alpha.

centuryControl

List that controls aspects of the Century model for carbon and nitrogen dynamics in the soil. It should be supplied through the centuryParms function.

`SC1-9` Soil carbon pools in the soil. SC1: Structural surface litter. SC2: Metabolic surface litter. SC3: Structural root litter. SC4: Metabolic root litter. SC5: Surface microbe. SC6: Soil microbe. SC7: Slow carbon. SC8: Passive carbon. SC9: Leached carbon.

`LeafL.Ln` Leaf litter lignin content.

`StemL.Ln` Stem litter lignin content.

`RootL.Ln` Root litter lignin content.

`RhizomeL.Ln` Rhizome litter lignin content.

`LeafL.N` Leaf litter nitrogen content.

`StemL.N` Stem litter nitrogen content.

`RootL.N` Root litter nitrogen content.

`RhizomeL.N` Rhizome litter nitrogen content.

`Nfert` Nitrogen from a fertilizer source.

`iMinN` Initial value for the mineral nitrogen pool.

`Litter` Initial values of litter (leaf, stem, root, rhizome).

`timestep` currently either week (default) or day.

Value

a `list` structure with components

Examples

```
## Not run:
data(weather05)

res0 <- BioGro(weather05)

plot(res0)

## Looking at the soil model

res1 <- BioGro(weather05, soilControl = soilParms(soilLayers = 6))
plot(res1, plot.kind='SW') ## Without hydraulic distribution
res2 <- BioGro(weather05, soilControl = soilParms(soilLayers = 6, hydrDist=TRUE))
plot(res2, plot.kind='SW') ## With hydraulic distribution


## Example of user defined soil parameters.
## The effect of phi2 on yield and soil water content

ll.0 <- soilParms(FieldC=0.37,WiltP=0.2,phi2=1)
ll.1 <- soilParms(FieldC=0.37,WiltP=0.2,phi2=2)
ll.2 <- soilParms(FieldC=0.37,WiltP=0.2,phi2=3)
ll.3 <- soilParms(FieldC=0.37,WiltP=0.2,phi2=4)

ans.0 <- BioGro(weather05,soilControl=ll.0)
ans.1 <- BioGro(weather05,soilControl=ll.1)
ans.2 <- BioGro(weather05,soilControl=ll.2)
```

```
ans.3 <-BioGro(weather05,soilControl=ll.3)

xyplot(ans.0$SoilWatCont +
       ans.1$SoilWatCont +
       ans.2$SoilWatCont +
       ans.3$SoilWatCont ~ ans.0$DayofYear,
       type='l',
       ylab='Soil water Content (fraction)',
       xlab='DOY')

## Compare LAI

xyplot(ans.0$LAI +
       ans.1$LAI +
       ans.2$LAI +
       ans.3$LAI ~ ans.0$DayofYear,
       type='l',
       ylab='Leaf Area Index',
       xlab='DOY')



## End(Not run)
```

`doy124` *Weather data*

Description

Example for a given day of the year to illustrate the `CanA` function.

Format

data frame of dimensions 24 by 8.

Details

LAI: leaf area index.

year: year.

doy: 124 in this case.

hour: hour of the day, (0–23).

solarR: direct solar radiation.

DailyTemp.C: hourly air temperature (Celsius).

RH: relative humidity, (0–1).

WindSpeed: 4.1 m s^{-1} average daily value in this case.

Source

simulated

`eC4photo` *C4 photosynthesis simulation (von Caemmerer model)*

Description

Simulation of C4 photosynthesis based on the equations proposed by von Caemmerer (2000). At this point assimilation and stomatal conductance are not coupled and although, for example a lower relative humidity will lower stomatal conductance it will not affect assimilation. Hopefully, this will be improved in the future.

Usage

```
eC4photo(Qp, airtemp, rh, ca = 380, oa = 210, vcmax = 60, vpmax = 120,
  vpr = 80, jmax = 400)
```

Arguments

`Qp`	quantum flux (μ mol m^{-2} s^{-1}).
`airtemp`	air temperature (Celsius).
`rh`	relative humidity in proportion (e.g. 0.7).
`ca`	atmospheric carbon dioxide concentration (ppm or μbar) (e.g. 380).
`oa`	atmospheric oxygen concentration (mbar) (e.g. 210).
`vcmax`	Maximum rubisco activity (μ mol m^{-2} s^{-1}).
`vpmax`	Maximum PEP carboxylase activity (μ mol m^{-2} s^{-1}).
`vpr`	PEP regeneration rate (μ mol m^{-2} s^{-1}).
`jmax`	Maximal electron transport rate (μmol electrons m^{-2} s^{-1}).

Details

The equations are taken from von Caemmerer (2000) for the assimilation part and stomatal conductance is based on FORTRAN code by Joe Berry (translated to C).

Value

results of call to C function eC4photo_sym

a `list` structure with components

References

Susanne von Caemmerer (2000) Biochemical Models of Leaf Photosynthesis. CSIRO Publishing. (In particular chapter 4).

Examples

```
## Not run:
## A simple example for the use of eC4photo
## This is the model based on von Caemmerer
## First we can compare the effect of varying
## Jmax. Notice that this is different from
## varying alpha in the Collatz model

Qps <- seq(0,2000,10)
Tls <- seq(0,55,5)
rhs <- c(0.7)
dat1 <- data.frame(expand.grid(Qp=Qps,Tl=Tls,RH=rhs))
res1 <- eC4photo(dat1$Qp,dat1$Tl,dat1$RH)
res2 <- eC4photo(dat1$Qp,dat1$Tl,dat1$RH,jmax=700)

## Plot comparing Jmax 400 vs. 700 for a range of conditions
xyplot(res1$Assim + res2$Assim ~ Qp | factor(Tl) , data = dat1,
            type='l',col=c('blue','green'),lwd=2,
            ylab=expression(paste('Assimilation (',
                 mu,mol,' ',m^-2,' ',s^-1,')')),
             xlab=expression(paste('Quantum flux (',
                 mu,mol,' ',m^-2,' ',s^-1,')')),
            key=list(text=list(c('Jmax 400','Jmax 700')),
              lines=TRUE,col=c('blue','green'),lwd=2))

## Second example is the effect of varying Vcmax

Qps <- seq(0,2000,10)
Tls <- seq(0,35,5)
rhs <- 0.7
vcmax <- seq(0,40,5)
dat1 <- data.frame(expand.grid(Qp=Qps,Tl=Tls,RH=rhs,vcmax=vcmax))
res1 <- numeric(nrow(dat1))
for(i in 1:nrow(dat1)){
res1[i] <- eC4photo(dat1$Qp[i],dat1$Tl[i],dat1$RH[i],vcmax=dat1$vcmax[i])$Assim
}

## Plot comparing different Vcmax
cols <- rev(heat.colors(9))
xyplot(res1 ~ Qp | factor(Tl) , data = dat1,col=cols,
            groups=vcmax,
            type='l',lwd=2,
            ylab=expression(paste('Assimilation (',
                 mu,mol,' ',m^-2,' ',s^-1,')')),
             xlab=expression(paste('Quantum flux (',
                 mu,mol,' ',m^-2,' ',s^-1,')')),
            key=list(text=list(as.character(vcmax)),
              lines=TRUE,col=cols,lwd=2))



## End(Not run)
```

eCanA *Simulates canopy assimilation (von Caemmerer model)*

Description

It represents an integration of the photosynthesis function `eC4photo`, canopy evapo/transpiration and the multilayer canopy model `sunML`.

Usage

```
eCanA(LAI, doy, hour, solarR, AirTemp, RH, WindS, Vcmax, Vpmax, Vpr, Jmax,
  Ca = 380, Oa = 210, StomataWS = 1)
```

Arguments

`LAI`	leaf area index.
`doy`	day of the year, (1–365).
`hour`	hour of the day, (0–23).
`solarR`	solar radiation ($\mu mol\ m^{-2}\ s^{-1}$).
`AirTemp`	temperature (Celsius).
`RH`	relative humidity (0–1).
`WindS`	wind speed ($m\ s^{-1}$).
`Vcmax`	Maximum rubisco activity ($\mu mol\ m^{-2}\ s^{-1}$).
`Vpmax`	Maximum PEP carboxylase activity ($\mu mol\ m^{-2}\ s^{-1}$).
`Vpr`	PEP regeneration rate ($\mu mol\ m^{-2}\ s^{-1}$).
`Jmax`	Maximal electron transport rate (μmol electrons $m^{-2}\ s^{-1}$).
`Ca`	atmospheric carbon dioxide concentration (ppm or μbar) (e.g. 380).
`Oa`	atmospheric oxygen concentration (mbar) (e.g. 210).
`StomataWS`	Effect of water stress on assimilation.

Value

`numeric`

returns a single value which is hourly canopy assimilation (mol m^{-2} ground hr^{-1})

Examples

```
## Not run:
data(doy124)
tmp1 <- numeric(24)
for(i in 1:24){
  lai <- doy124[i,1]
  doy <- doy124[i,3]
  hr  <- doy124[i,4]
```

```
  solar <- doy124[i,5]
  temp <- doy124[i,6]
  rh <- doy124[i,7]/100
  ws <- doy124[i,8]

  tmp1[i] <- CanA(lai,doy,hr,solar,temp,rh,ws)
}

plot(c(0:23),tmp1,
     type='l',lwd=2,
     xlab='Hour',
     ylab=expression(paste('Canopy assimilation (mol  ',
     m^-2,' ',s^-1,')')))

## End(Not run)
```

`EngWea94i` *Weather data corresponding to a paper by Clive Beale (see source).*

Description

Weather data with the precipitation column giving precipitation plus irrigation.

Format

A data frame with 8760 observations on the following 8 variables.

list('year') a numeric vector

list('doy') a numeric vector

list('hour') a numeric vector

list('solarR') a numeric vector

list('DailyTemp.C') a numeric vector

list('RH') a numeric vector

list('WindSpeed') a numeric vector

list('precip') a numeric vector

Details

~~ If necessary, more details than the above ~~

Source

~~ reference to a publication or URL from which the data were obtained ~~

References

~~ possibly secondary sources and usages ~~

Examples

```
data(EngWea94i)
## maybe str(EngWea94i) ; plot(EngWea94i) ...
```

EngWea94rf *Weather data corresponding to a paper by Clive Beale (see source).*

Description

Weather data with the precipitation column giving precipitation without irrigation.

Format

A data frame with 8760 observations on the following 8 variables.

list('year') a numeric vector

list('doy') a numeric vector

list('hour') a numeric vector

list('solarR') a numeric vector

list('DailyTemp.C') a numeric vector

list('RH') a numeric vector

list('WindSpeed') a numeric vector

list('precip') a numeric vector

Details

~~ If necessary, more details than the description above ~~

Source

~~ reference to a publication or URL from which the data were obtained ~~

References

~~ possibly secondary sources and usages ~~

Examples

```
data(EngWea94rf)
## maybe str(EngWea94rf) ; plot(EngWea94rf) ...
```

flow — *Returns values based on if kno is less than, equal to, or greater than three.*

Description

Returns Values for SC, fC, Resp, Kf, and MinN to be used in the Century function

Usage

```
flow(SC, CNratio, A, Lc, Tm, resp, kno, Ks, verbose = FALSE)
```

Arguments

SC	Soil Carbon
CNratio	ratio of carbon to nitrogen
A	effects of teperature and moisture
Lc	See FmLcFun
TM	effect of soil texture on active SOM turnover
resp	respiration
kno	an integer value which determines
Ks	flow constant
verbose	Only used in the R version for debugging

FmLcFun — *Returns values for Fm and Lc*

Description

A basic function designed to define the value for Fm and Lc which are used in the century function.

Usage

```
FmLcFun(Lig, Nit)
```

Arguments

Lig	lignin
Nit	nitrogen

`fnpsvp` *The Goff Gratch equation from Smithsonian Tables, 1984. http://cires.colorado.edu/~voemel/vp.html*

Description

The Goff Gratch equation from Smithsonian Tables, 1984. http://cires.colorado.edu/~voemel/vp.html

Usage

```
fnpsvp(Tkelvin)
```

Arguments

`Tkelvin`	the abolute temperature

`idbp` *Initial Dry Biomass Partitioning Coefficients*

Description

Atempts to guess good initial vales for dry biomass coefficients that can be passed to `BioGro`, `OpBioGro`, `constrOpBioGro`, or `MCMCBioGro`. It is very fragile.

Usage

```
idbp(data, phenoControl = list())
```

Arguments

`data`	Should have at least five columns with: ThermalT, Stem, Leaf, Root, Rhizome and Grain.
`phenoControl`	list that supplies mainly in this case the thrmal time periods that delimit the phenological stages,

Details

This function will not accept missing values. It can be quite fragile and it is rather inflexible in what it expects in terms of data.

Value

It returns a vector of length 25 suitable for `BioGro`, `OpBioGro`, `constrOpBioGro`, or `MCMCBioGro`.

Note

It is highly recommended that the results of this function are tested with `valid_dbp`.

Author(s)

Fernando E. Miguez

See Also

`valid_dbp`

Examples

```
## See ?OpBioGro
```

`idbpm` *idpm*

Description

This estimates initial dry biomass partitioning coefficients based on data for an annual grass

Usage

```
idbpm(data, MaizePhenoControl = list())
```

Arguments

`data`	data frame ThermalT Stem Leaf Root Grain LAI 1 0.211 0.00733 0.00104 0.00704 0 0.00119 611 280.000 1.08019 0.95531 0.11618 0 1.62350 1221 560.000 5.91862 2.08684 1.42061 0 3.54748 1831 747.000 10.16707 2.36378 2.53192 0 4.01843 2442 969.000 15.08485 2.42849 3.57410 0 4.12843 3052 1080.000 18.56392 2.46765 4.32115 0 4.19501 3662 1136.000 20.87121 2.04021 4.82178 0 3.46836 4273 1452.000 22.05770 0.89954 5.20210 0 1.52921
`MaizePhenoControl`	

Value

vector of biomass pools

Author(s)

Fernando E. Miguez

LayET *Weather data*

Description

Layer data for evapo/transpiration. Simulated data to show the result of the EvapoTrans function.

Format

data frame of dimensions 384 by 9.

Details

lfClass: leaf class, 'sun' or 'shade'.

layer: layer in the canopy, 1 to 8.

hour: hour of the day, (0–23).

Rad: direct light.

Itot: total radiation.

Temp: air temperature, (Celsius).

RH: relative humidity, (0–1).

WindSpeed: wind speed, (m s^{-1}).

LAI: leaf area index.

Source

simulated

lightME *Simulates the light macro environment*

Description

Simulates light macro environment based on latitude, day of the year. Other coefficients can be adjusted.

Usage

```
lightME(lat = 40, DOY = 190, t.d = 12, t.sn = 12, atm.P = 1e+05,
  alpha = 0.85)
```

Arguments

lat	the latitude, default is 40 (Urbana, IL, U.S.).
DOY	the day of the year (1–365), default 190.
t.d	time of the day in hours (0–23), default 12.
t.sn	time of solar noon, default 12.
atm.P	atmospheric pressure, default 1e5 (kPa).
alpha	atmospheric transmittance, default 0.85.

Value

a `list` structure with components

Examples

```
## Direct and diffuse radiation for DOY 190 and hours 0 to 23

res <- lightME(t.d=0:23)

xyplot(I.dir + I.diff ~ 0:23 , data = res,
type='o',xlab='hour',ylab='Irradiance')

xyplot(propIdir + propIdiff ~ 0:23 , data = res,
type='o',xlab='hour',ylab='Irradiance proportion')
```

MaizeGro *Simulation of Maize, Growth, LAI, Photosynthesis and phenology*

Description

It takes weather data as input (hourly timesteps) and several parameters and it produces phenology, photosynthesis, LAI, etc.

Usage

```
MaizeGro(WetDat, plant.day = NULL, emerge.day = NULL, harvest.day = NULL,
  plant.density = 7, timestep = 1, lat = 40, canopyControl = list(),
  MaizeSeneControl = list(), photoControl = list(),
  MaizePhenoControl = list(), MaizeCAllocControl = list(),
  laiControl = list(), soilControl = list(), MaizeNitroControl = list(),
  centuryControl = list())
```

Arguments

WetDat	weather data as produced by the weach function.
plant.day	Planting date (format 0-365)
emerge.day	Emergence date (format 0-365)
harvest.day	Harvest date (format 0-365)
plant.density	Planting density (plants per meter squared, default = 7)
timestep	Simulation timestep, the default of 1 requires houlry weather data. A value of 3 would require weather data every 3 hours. This number should be a divisor of 24.
lat	latitude, default 40.
canopyControl	List that controls aspects of the canopy simulation. It should be supplied through the canopyParms function. Sp (specific leaf area) here the units are ha Mg^{-1}. If you have data in m^2 of leaf per kg of dry matter (e.g. 15) then divide by 10 before inputting this coefficient. SpD decrease of specific leaf area. Empirical parameter. Default 0. example value (1.7e-3). nlayers (number of layers of the canopy) Maximum 50. To increase the number of layers (more than 50) the C source code needs to be changed slightly. kd (extinction coefficient for diffuse light) between 0 and 1. mResp (maintenance respiration) a vector of length 2 with the first component for leaf and stem and the second component for rhizome and root.
MaizeSeneControl	List that controls aspects of senescence simulation. It should be supplied through the MaizeSeneParms function. senLeaf Thermal time at which leaf senescence will start. senStem Thermal time at which stem senescence will start. senRoot Thermal time at which root senescence will start.
photoControl	List that controls aspects of photosynthesis simulation. It should be supplied through the MaizePhotoParms function. vmax Vmax passed to the c4photo function. alpha alpha parameter passed to the c4photo function. kparm kparm parameter passed to the c4photo function. theta theta parameter passed to the c4photo function. beta beta parameter passed to the c4photo function. Rd Rd parameter passed to the c4photo function. UPPERTEMP UPPERTEMP parameter passed to the c4photo function. LOWERTEMP LOWERTEMP parameter passed to the c4photo function. Catm Catm parameter passed to the c4photo function. b0 b0 parameter passed to the c4photo function. b1 b1 parameter passed to the c4photo function.

`MaizePhenoControl`
: argument used to pass parameters related to phenology characteristics `MaizePhenoControl` here~~

`soilControl`
: here~~

`nitroControl`
: here~~

`centuryControl`
: here~~

Details

The phenology follows the 'Corn Growth and Development' Iowa State Publication. than the description above ~~

Value

It currently returns a list with the following components

`DayofYear`	Day of the year (0-365)
`Hour`	Hour of the day (0-23)
`TTTc`	Accumulated thermal time
`PhenoStage`	Phenological stage of the crop
`CanopyAssim`	Hourly canopy assimilation, (Mg ha^-1 ground hr^-1).
`CanopyTrans`	Hourly canopy transpiration, (Mg ha^-1 ground hr^-1).
`LAI`	Leaf Area Index

Author(s)

Fernando E Miguez

See Also

`BioGro` `help`, ~~~

Examples

```
data(weather05)
res <- MaizeGro(weather05, plant.day = 110, emerge.day = 120, harvest.day=300,
                  MaizePhenoControl = MaizePhenoParms(R6 = 2000))
```

MaizePhenoParms *Lists values for phenology parameters*

Description

Lists values for phenology parameters

Usage

```
MaizePhenoParms(base.temp = 10, max.leaves = 20, plant.emerg = 100,
  phyllochron1 = 46.7, phyllochron2 = 31.1, R1 = 747, R2 = 858,
  R3 = 969, R4 = 1080, R5 = 1136, R6 = 1452)
```

Arguments

base.temp	inital temperatre
max.leaves	maximum leaves
plant.emerg	plant emergence
phyllochron1	needs description
phyllochron2	needs description
R1	needs description
R2	needs description
R3	needs description
R4	needs description
R5	needs description
R6	needs description

MaizeSeneParms *Lists values for senecence parameters*

Description

Lists values for senecence parameters

Usage

```
MaizeSeneParms(senStem = 3000, senLeaf = 3500, senRoot = 4000)
```

Arguments

senStem	senecence of stem
senLeaf	senecence of leaf
senRoot	senecence of root

MCMCBioGro *Simulated annealing and MCMC function*

Description

Simulated Annealing and Markov chain Monte Carlo for estimating parameters for Biomass Growth

Usage

```
MCMCBioGro(niter = 10, niter2 = 10, phen = 6, iCoef = NULL,
  saTemp = 5, coolSamp = 20, scale = 0.5, WetDat, data, day1 = NULL,
  dayn = NULL, timestep = 1, lat = 40, iRhizome = 7, irtl = 1e-04,
  canopyControl = list(), seneControl = list(), photoControl = list(),
  phenoControl = list(), soilControl = list(), nitroControl = list(),
  centuryControl = list(), sd = c(0.02, 1e-06))
```

Arguments

niter	number of iterations for the simulated annealing portion of the optimization.
niter2	number of iterations for the Markov chain Monte Carlo portion of the optimization.
phen	Phenological stage being optimized.
iCoef	initial coefficients for dry biomass partitioning.
saTemp	simulated annealing temperature.
coolSamp	number of cooling samples.
scale	scale parameter to control the standard deviations.
WetDat	weather data.
data	observed data.
day1	first day of the growing season.
dayn	last day of the growing season.
timestep	Timestep see BioGro.
lat	latitude.
iRhizome	initial rhizome biomass.
irtl	See BioGro.
canopyControl	See canopyParms.
seneControl	See seneParms.
photoControl	See photoParms.
phenoControl	See phenoParms.
soilControl	See soilParms.
nitroControl	See nitroParms.

centuryControl
: See centuryParms.

sd
: standard deviations for the parameters to be optimized. The first (0.02) is for the positive dry biomass partitioning coefficients. The second (1e-6) is for the negative dry biomass partitioning coefficients.

Details

This function atempts to implement the simulated annealing method for estimating parameters of a generic C4 crop growth.

This function implements a hybrid algorithm where the first portion is simulated annealing and the second portion is a Markov chain Monte Carlo. The user controls the number of iterations in each portion of the chain with niter and niter2.

Value

An object of class MCMCBioGro consisting of a list with 23 components. The easiest way of accessing the information is with the print and plot methods.

Note

The automatic method for guessing the last day of the growing season differs slightly from that in BioGro. To prevent error due to a shorter simulated growing season than the observed one the method in MCMCBioGro adds one day to the last day of the growing season. Although the upper limit is fixed at 330.

Author(s)

Fernando E. Miguez

Fernando E. Miguez

See Also

See Also as BioGro, OpBioGro and constrOpBioGro.

Examples

```
## Not run:

data(weather05)

## Some coefficients
pheno.ll <- phenoParms(kLeaf1=0.48,kStem1=0.47,kRoot1=0.05,kRhizome1=-1e-4,
                       kLeaf2=0.14,kStem2=0.65,kRoot2=0.21, kRhizome2=-1e-4,
                       kLeaf3=0.01, kStem3=0.56, kRoot3=0.13, kRhizome3=0.3,
                       kLeaf4=0.01, kStem4=0.56, kRoot4=0.13, kRhizome4=0.3,
                       kLeaf5=0.01, kStem5=0.56, kRoot5=0.13, kRhizome5=0.3,
                       kLeaf6=0.01, kStem6=0.56, kRoot6=0.13, kRhizome6=0.3)

system.time(ans <- BioGro(weather05, phenoControl = pheno.ll))
```

```
ans.dat <- as.data.frame(unclass(ans)[1:11])
sel.rows <- seq(1,nrow(ans.dat),400)
simDat <- ans.dat[sel.rows,c('ThermalT','Stem','Leaf','Root','Rhizome','Grain','LAI')]
plot(ans,simDat)

## Residual sum of squares before the optimization

ans0 <- BioGro(weather05)
RssBioGro(simDat,ans0)


op1.mc <- MCMCBioGro(phen=1, niter=200,niter2=200,
                     WetDat=weather05,
                     data=simDat)


plot(op1.mc)

plot(op1.mc, plot.kind='trace', subset = nams %in%
\t\t\t\tc('kLeaf_1','kStem_1','kRoot_1'))


## End(Not run)
```

`MCMCc4photo` *Markov chain Monte Carlo for C4 photosynthesis parameters*

Description

This function implement Markov chain Monte Carlo methods for the C4 photosynthesis model of Collatz et al. The chain is constructed using a Gaussian random walk. This is definitely a beta version of this function and more testing and improvements are needed. The value of this function is in the possibility of examining the empirical posterior distribution (i.e. vectors) of the vmax and alpha parameters. Notice that you will get different results each time you run it.

Usage

```
MCMCc4photo(data, niter = 20000, ivmax = 39, ialpha = 0.04,
  ikparm = 0.7, itheta = 0.83, ibeta = 0.93, iRd = 0.8, Catm = 380,
  b0 = 0.08, b1 = 3, StomWS = 1, ws = c("gs", "vmax"), scale = 1,
  sds = c(1, 0.005), prior = c(39, 10, 0.04, 0.02), UPPERTEMP = 37.5,
  LOWERTEMP = 3)
```

Arguments

`data`	observed assimilation data, which should be a data frame or matrix. The first column should be observed net assimilation rate ($\mu mol\ m^{-2}\ s^{-1}$). The second

	column should be the observed quantum flux ($\mu mol\ m^{-2}\ s^{-1}$). The third column should be observed temperature of the leaf (Celsius). The fourth column should be the observed relative humidity in proportion (e.g. 0.7).
niter	number of iterations to run the chain for (default = 20000).
ivmax	initial value for Vcmax (default = 39).
ialpha	initial value for alpha (default = 0.04).
ikparm	initial value for kparm (default = 0.7). Not optimized at the moment.
itheta	initial value for theta (default = 0.83). Not optimized at the moment.
ibeta	initial value for beta (default = 0.93). Not optimized at the moment.
iRd	initial value for dark respiration (default = 0.8).
Catm	see c4photo function.
b0	see c4photo function.
b1	see c4photo function.
StomWS	see c4photo function.
ws	see c4photo function.
scale	This scale parameter controls the size of the standard deviations which generate the moves in the chain.
sds	Finer control for the standard deviations of the prior normals. The first element is for vmax and the second for alpha.
prior	Vector of length 4 with first element prior mean for vmax, second element prior standard deviation for vmax, third element prior mean for alpha and fourth element prior standard deviation for alpha.

Value

an object of class MCMCc4photo with components

References

Brooks, Stephen. (1998). Markov chain Monte Carlo and its application. The Statistician. 47, Part 1, pp. 69-100.

Examples

```
## Not run:
## Using Beale, Bint and Long (1996)
data(obsBea)

## Different starting values
resB1 <- MCMCc4photo(obsBea, 100000, scale=1.5)
resB2 <- MCMCc4photo(obsBea, 100000, ivmax=25, ialpha=0.1, scale=1.5)
resB3 <- MCMCc4photo(obsBea, 100000, ivmax=45, ialpha=0.02, scale=1.5)

## Use the plot function to examine results
plot(resB1,resB2,resB3)
```

```
plot(resB1,resB2,resB3,plot.kind='density',burnin=1e4)

## End(Not run)
```

`MCMCEc4photo` *Markov chain Monte Carlo for C4 photosynthesis parameters*

Description

This function attempts to implement Markov chain Monte Carlo methods for models with no likelihoods. In this case it is done for the von Caemmerer C4 photosynthesis model. The method implemented is based on a paper by Marjoram et al. (2003). The method is described in Miguez (2007). The chain is constructed using a Gaussian random walk. This is definitely a beta version of this function and more testing and improvements are needed. The value of this function is in the possibility of examining the empirical posterior distribution (i.e. vectors) of the Vcmax and alpha parameters. Notice that you will get different results each time you run it.

Usage

```
MCMCEc4photo(obsDat, niter = 30000, iCa = 380, iOa = 210, iVcmax = 60,
  iVpmax = 120, iVpr = 80, iJmax = 400, thresh = 0.01, scale = 1)
```

Arguments

`obsDat`	observed assimilation data, which should be a data frame or matrix. The first column should be observed net assimilation rate (μ mol m^{-2} s^{-1}). The second column should be the observed quantum flux (μ mol m^{-2} s^{-1}). The third column should be observed temperature of the leaf (Celsius). The fourth column should be the observed relative humidity in proportion (e.g. 0.7).
`niter`	number of iterations to run the chain for (default = 30000).
`iCa`	CO2 atmospheric concentration (ppm or μbar).
`iOa`	O2 atmospheric concentration (mbar).
`iVcmax`	initial value for Vcmax (default = 60).
`iVpmax`	initial value for Vpmax (default = 120).
`iVpr`	initial value for Vpr (default = 80).
`iJmax`	initial value for Jmax (default = 400).
`thresh`	this is a threshold that determines the "convergence" of the initial burn-in period. The convergence of the whole chain can be evaluated by running the model with different starting values for Vcmax and alpha. The chain should convergence, but for this, runs with similar thresholds should be compared. This threshold reflects the fact that for any given data the model will not be able to simulate it perfectly so it represents a compromise between computability and fit.
`scale`	This scale parameter controls the size of the standard deviations which generate the moves in the chain. Decrease it to increase the acceptance rate and viceversa.

Value

a `list` structure with components

References

P. Marjoram, J. Molitor, V. Plagnol, S. Tavare, Markov chain monte carlo without likelihoods, PNAS 100 (26) (2003) 15324–15328.

Miguez (2007) Miscanthus x giganteus production: meta-analysis, field study and mathematical modeling. PhD Thesis. University of Illinois at Urbana-Champaign.

Examples

```
## Not run:
## This is an example for the MCMCEc4photo
## evaluating the convergence of the chain
## Notice that if a parameter does not seem
## to converge this does not mean that the method
## doesn't work. Careful examination is needed
## in order to evaluate the validity of the results
data(obsNaid)
res1 <- MCMCEc4photo(obsNaid,100000,thresh=0.007)
res1

## Run it a few more times
## and test the stability of the method
res2 <- MCMCEc4photo(obsNaid,100000,thresh=0.007)
res3 <- MCMCEc4photo(obsNaid,100000,thresh=0.007)

## Now plot it
plot(res1,res2,res3)
plot(res1,res2,res3,type='density')

## End(Not run)
```

---

mOpc3photo — *Multiple optimization of assimilation (or stomatal conductance) curves.*

Description

It is a wrapper for Opc3photo which allows for optimization of multiple runs of curves (A/Q or A/Ci).

Usage

```
mOpc3photo(data, ID = NULL, iVcmax = 100, iJmax = 180, iRd = 1.1,
  op.level = 1, curve.kind = c("Ci", "Q"), verbose = FALSE, ...)
```

Arguments

data	should be a data.frame or matrix with x columns col 1: should be an ID for the different runs col 2: measured assimilation (CO2 uptake) col 3: Incomming PAR (photosynthetic active radiation) col 4: Leaf temperature col 5: Relative humidity col 6: Intercellular CO2 (for A/Ci curves) col 7: Reference CO2 level
ID	optional argument to include ids. should be of length equal to the number of runs.
iVcmax	Single value or vector of length equal to number of runs to supply starting values for the optimization of vcmax.
iJmax	Single value or vector of length equal to number of runs to supply starting values for the optimization of jmax.
iRd	Single value or vector of length equal to number of runs to supply starting values for the optimization of Rd.
op.level	Level 1 will optimize Vcmax and Jmax and level 2 will optimize Vcmax, Jmax and Rd.
curve.kind	Whether an A/Ci curve is being optimized or an A/Q curve.
verbose	Whether to print information about progress.
...	Additional arguments to be passed to Opc3photo

Details

Include more details about the data.

Value

an object of class 'mOpc3photo'

if op.level equals 1 best Vcmax, Jmax and convergence

if op.level equals 2 best Vcmax, Jmax, Rd and convergence

comp1	Description of 'comp1'

Author(s)

Fernando E. Miguez

See Also

See also Opc3photo as help, ~~~

Examples

```
data(simAssim)
simAssim <- cbind(simAssim[,1:6],Catm=simAssim[,10])
simAssim <- simAssim[simAssim[,1] < 11,]

plotAC(simAssim, trt.col=1)

op.all <- mOpc3photo(simAssim, op.level=1,
                     verbose=TRUE)

plot(op.all)
plot(op.all, parm='jmax')
```

`mOpc4photo` *Multiple optimization of C4 photosynthesis.*

Description

Wrapper function that allows for optimization of multiple A/Ci or A/Q curves.

Usage

```
mOpc4photo(data, ID = NULL, ivmax = 39, ialpha = 0.04, iRd = 0.8,
  iupperT = 37.5, ilowerT = 3, ikparm = 0.7, itheta = 0.83,
  ibeta = 0.93, curve.kind = c("Q", "Ci"), op.level = 1, op.ci = FALSE,
  verbose = FALSE, ...)
```

Arguments

`data`	observed assimilation data, which should be a data frame or matrix. The first column should contain a run or id. The second column should be observed net assimilation rate (μ mol m^{-2} s^{-1}). The third column should be the observed quantum flux (μ mol m^{-2} s^{-1}). The fourth column should be observed temperature of the leaf (Celsius). The fifth column should be the observed relative humidity in proportion (e.g. 0.7). An optional sixth column can contain atmospheric CO2.
`ID`	Optional vector with an alternative ID tothe one used in data for runs. The length shoudl be equal to the number of runs.
`ivmax`	Initial value for vmax. It can be a single value or a vector of length equal to the number of runs.
`ialpha`	Initial value for alpha. It can be a single value or a vector of length equal to the number of runs.
`iRd`	Initial value for vmax. It can be a single value or a vector of length equal to the number of runs.
`ikparm`	Initial value for vmax. It can be a single value or a vector of length equal to the number of runs.

itheta	Initial value for vmax. It can be a single value or a vector of length equal to the number of runs.
ibeta	Initial value for vmax. It can be a single value or a vector of length equal to the number of runs.
curve.kind	If 'Q' a type of response which mainly depends on light will be assumed. Typically used to optimized light response curves or diurnals. Use 'Ci' for A/Ci curves (stomatal conductance could also be optimized).
op.level	optimization level. If equal to 1 vmax and alpha will be optimized. If 2, vmax, alpha and Rd will be optimized. If 3, vmax, alpha, theta and Rd will be optimized.
op.ci	Whether to optimize intercellular CO2. Default is FALSE as 'fast-measured' light curves do not provide reliable values of Ci.
verbose	Whether to display output about convergence of each run.
...	Used to supply additional arguments to Opc4photo.

Details

There are printing and plotting methods for the object created by this function. The plotting function has an argument that it is used to dsiplay either vmax or alpha (i.e. parm=c('vmax','alpha')). In both cases the optimized value plus confidence intervals will be displayed for each run.

Value

An object of class mOpc4photo returned

Author(s)

Fernando E. miguez

See Also

Opc4photo c4photo optim help, ~~~

Examples

```
data(simAssim)
```

---

obsBea *Miscanthus assimilation field data*

Description

assimilation as measured in Beale, Bint and Long (1996) in Miscanthus. The first column is the observed net assimilation rate (μ mol m^{-2} s^{-1}). The second column is the observed quantum flux (μ mol m^{-2} s^{-1}). The third column is the temperature (Celsius).Relative humidity was not reported and thus was assumed to be 0.7.

Format

data frame of dimensions 27 by 4.

Source

C. V. Beale, D. A. Bint, S. P. Long, Leaf photosynthesis in the C4-grass miscanthus x giganteus, growing in the cool temperate climate of southern England, *J. Exp. Bot.* 47 (2) (1996) 267–273.

---

obsBeaC *Complete Miscanthus assimilation field data*

Description

Assimilation and stomatal conductance as measured in Beale, Bint and Long (1996) in Miscanthus (missing data are also included). The first column is the date, the second the hour. Columns 3 and 4 are assimilation and stomatal conductance respectively.

Format

data frame of dimensions 35 by 6.

Details

The third column is the observed net assimilation rate (μ mol m^{-2} s^{-1}).

The fifth column is the observed quantum flux (μ mol m^{-2} s^{-1}).

The sixth column is the temperature (Celsius).

Source

C. V. Beale, D. A. Bint, S. P. Long, Leaf photosynthesis in the C4-grass miscanthus x giganteus, growing in the cool temperate climate of southern England, *J. Exp. Bot.* 47 (2) (1996) 267–273.

---

obsNaid *Miscanthus assimilation data*

Description

assimilation as measured in Naidu et al. (2003) in Miscanthus. The first column is the observed net assimilation rate (μ mol m^{-2} s^{-1}) The second column is the observed quantum flux (μ mol m^{-2} s^{-1}) The third column is the temperature (Celsius). The fourth column is the observed relative humidity in proportion (e.g. 0.7).

Format

data frame of dimensions 16 by 4.

Source

S. L. Naidu, S. P. Moose, A. K. AL-Shoaibi, C. A. Raines, S. P. Long, Cold Tolerance of C4 photosynthesis in Miscanthus x giganteus: Adaptation in Amounts and Sequence of C4 Photosynthetic Enzymes, *Plant Physiol.* 132 (3) (2003) 1688–1697.

`OpBioGro` *Optimization of dry biomass partitioning coefficients.*

Description

Optimizes dry biomass partitioning coefficients using constrained optimization (see below).

Usage

```
OpBioGro(phen = 1, iCoef = NULL, WetDat, data, day1 = NULL, dayn = NULL,
  timestep = 1, lat = 40, iRhizome = 7, irtl = 1e-04,
  canopyControl = list(), seneControl = list(), photoControl = list(),
  phenoControl = list(), soilControl = list(), nitroControl = list(),
  centuryControl = list(), op.method = c("optim", "nlminb"),
  verbose = FALSE, ...)
```

Arguments

`phen`	integer taking values 1 through 6 which indicate the phenological stage being optimized. If all of the phenological stages need to be optimized then use zero (0).
`iCoef`	initial vector of size 24 for the dry biomass partitioning coefficients.
`WetDat`	Weather data.
`data`	observed data.
`day1`	first day of the growing season.
`dayn`	last day of the growing season.
`timestep`	see `BioGro`
`lat`	see `BioGro`
`iRhizome`	see `BioGro`
`irtl`	see `BioGro`
`canopyControl`	see `BioGro`
`seneControl`	see `BioGro`
`photoControl`	see `BioGro`
`phenoControl`	see `BioGro`
`soilControl`	see `BioGro`
`nitroControl`	see `BioGro`

centuryControl	see BioGro
op.method	Optimization method. Whether to use optim or nlminb
verbose	Displays additional information, originally used for debugging.
...	additional arguments passed to optim or constrOptim.

Details

The optimization is done over the BioGro function. The OpBioGro function is a wrapper for optim and the constrOpBioGro is a wrapper for constrOptim.

Value

list of class OpBioGro with components

Warning

This function has not had enough testing.

References

no references yet.

See Also

BioGro constrOptim optim c4photo

Examples

```
## Not run:

data(weather05)

## Some coefficients
pheno.ll <- phenoParms(kLeaf1=0.48,kStem1=0.47,kRoot1=0.05,kRhizome1=-1e-4,
                       kLeaf2=0.14,kStem2=0.65,kRoot2=0.21, kRhizome2=-1e-4,
                       kLeaf3=0.01, kStem3=0.56, kRoot3=0.13, kRhizome3=0.3,
                       kLeaf4=0.01, kStem4=0.56, kRoot4=0.13, kRhizome4=0.3,
                       kLeaf5=0.01, kStem5=0.56, kRoot5=0.13, kRhizome5=0.3,
                       kLeaf6=0.01, kStem6=0.56, kRoot6=0.13, kRhizome6=0.3)

system.time(ans <- BioGro(weather05, phenoControl = pheno.ll))

ans.dat <- as.data.frame(unclass(ans)[1:11])
sel.rows <- seq(1,nrow(ans.dat),length.out=8)
simDat <- ans.dat[sel.rows,c('ThermalT','Stem','Leaf','Root','Rhizome','Grain','LAI')]
plot(ans,simDat)

## Residual sum of squares before the optimization

ans0 <- BioGro(weather05)
```

```
RssBioGro(simDat,ans0)

## This will optimize only the first phenological stage
idb <- valid_dbp(idbp(simDat))
op1 <- OpBioGro(phen=0, WetDat=weather05, data = simDat, iCoef=idb)
## or
cop1 <- constrOpBioGro(phen=0, WetDat=weather05, data = simDat)


## End(Not run)
```

`Opc3photo` *Optimize parameters of the C3 photosynthesis model.*

Description

Applies the `optim` function to C3 photosynthesis.

Usage

```
Opc3photo(data, ivcmax = 100, ijmax = 180, iRd = 1.1, Catm = 380,
  O2 = 210, ib0 = 0.08, ib1 = 9.58, itheta = 0.7, op.level = 1,
  op.method = c("optim", "nlminb"), response = c("Assim", "StomCond"),
  level = 0.95, hessian = TRUE, curve.kind = c("Ci", "Q"),
  op.ci = FALSE, ...)
```

Arguments

`data`	should be a `data.frame` or `matrix` with x columns col 1: measured assimilation (CO2 uptake) col 2: Incomming PAR (photosynthetic active radiation) col 3: Leaf temperature col 4: Relative humidity col 5: Intercellular CO2 (for A/Ci curves) col 6: Reference CO2 level
`ivcmax`	Initial value for `vcmax`.
`ijmax`	Initial value for `jmax`.
`iRd`	Initial value for `Rd`.
`Catm`	Reference CO2.
`O2`	Reference level of O2.
`ib0`	Initial value for the intercept to the Ball-Berry model.
`ib1`	Initial value for the slope to the Ball-Berry model.
`itheta`	Initial value for `theta`.
`op.level`	Level 1 will optimize `Vcmax` and `Jmax` and level 2 will optimize `Vcmax`, `Jmax` and `Rd`.
`op.method`	optimization method. At the moment only optim is implemented.
`response`	`'Assim'` for assimilation and `'StomCond'` for stomatal conductance.

level	Confidence interval level.
hessian	Whether the hessian should be computed
curve.kind	Whether an A/Ci curve is being optimized or an A/Q curve.
op.ci	whether to optimize intercellular CO2.
...	Additioanl arguments to be passed to optim.

Value

An object of class Opc3photo.

The following components can be extracted:

Note

~~further notes~~ Additional notes about the assumptions.

Author(s)

Fernando E. Miguez

See Also

See Also mOpc3photo

Examples

```
## Load fabricated data
data(simA100)
## Look at it
head(simA100)

op <- Opc3photo(simA100[,1:5],Catm=simA100[,9], op.level = 2)

## If faced with a difficult problem
## This can give starting values
op100 <- Opc3photo(simA100[,1:5],Catm=simA100[,9],
                   op.level = 2, method='SANN',
                   hessian=FALSE)

op100 <- Opc3photo(simA100[,1:5],Catm = simA100[,9],
                   op.level = 2,
                   ivcmax = op100$bestVmax,
                   ijmax = op100$bestJmax,
                   iRd = op100$bestRd)
op100
```

`Opc4photo` *Optimization of C4 photosynthesis parameters*

Description

Optimization method for the Collatz C4 photosynthesis model. At the moment Vcmax and alpha are optimized only.

Usage

```
Opc4photo(data, ivmax = 39, ialpha = 0.04, iRd = 0.8, ikparm = 0.7,
  itheta = 0.83, ibeta = 0.93, Catm = 380, ib0 = 0.08, ib1 = 3,
  iStomWS = 1, ws = c("gs", "vmax"), iupperT = 37.5, ilowerT = 3,
  response = c("Assim", "StomCond"), curve.kind = c("Q", "Ci"),
  op.level = 1, level = 0.95, hessian = TRUE, op.ci = FALSE, ...)
```

Arguments

`data`	observed assimilation data, which should be a data frame or matrix. The first column should be observed net assimilation rate (μ mol m^{-2} s^{-1}). The second column should be the observed quantum flux (μ mol m^{-2} s^{-1}). The third column should be observed temperature of the leaf (Celsius). The fourth column should be the observed relative humidity in proportion (e.g. 0.7). An optional fifth column can contain intercellular CO2. The reference level of CO2 should be supplied to the function using the `Catm` argument.
`ivmax`	initial value for Vcmax (default = 39).
`ialpha`	initial value for alpha (default = 0.04).
`iRd`	initial value for dark respiration (default = 0.8).
`ikparm`	initial value for k (default = 0.7).
`itheta`	initial value for theta (default = 0.83).
`ibeta`	initial value for beta (default = 0.93).
`Catm`	Atmospheric CO2 in ppm (or μmol/mol).
`ib0`	initial value for the Ball-Berry intercept.
`ib1`	initial value for the Ball-Berry slope.
`iStomWS`	initial value for the stomata water stress factor.
`ws`	`ws` flag. See `c4photo`.
`response`	Use `'Assim'` if you want to optimize assimilation data and use `'StomCond'` if you want to optimize stomatal conductance data. The parameters optimized will be different.
`curve.kind`	If `'Q'` a type of response which mainly depends on light will be assumed. Typically used to optimized light response curves or diurnals. Use `'Ci'` for A/Ci curves (stomatal conductance could also be optimized).

op.level	optimization level. If equal to 1 vmax and alpha will be optimized. If 2, vmax, alpha and Rd will be optimized. If 3, vmax, alpha, theta and Rd will be optimized.
level	level for the confidence intervals.
hessian	Whether the hessian matrix should be computed. See optim.
op.ci	Whether to optimize intercellular CO2. Default is FALSE as 'fast-measured' light curves do not provide reliable values of Ci.
list()	Additional arguments passed to the optim in particular if a lower or upper bound is desired this could be achieved by adding lower=c(0,0) this will impose a lower bound on vmax and alpha of zero so preventing negative values from being returned. When the lower argument is added the optimization method changes from Nelder-Mead to BFGS.

Value

An object of class Opc4photo a list with components

If op.level 2 bestRd will also be supplied. If op.level 3 theta and bestRd will also be supplied.

If op.level 2 ciRd will also be supplied. If op.level 3 ciTheta and ciRd will also be supplied.

See Also

c4photo optim

Examples

```
data(aq)
## Select data for a single AQ optimization
aqd <- data.frame(aq[aq[,1] == 6,-c(1:2)],Catm=400)
res <- Opc4photo(aqd, Catm=aqd$Catm)
res

plot(res, plot.kind = 'OandF', type='o')
```

OpEC4photo — *Optimization of C4 photosynthesis parameters (von Caemmerer model)*

Description

Optimization method for the von Caemmerer C4 photosynthesis model.

Usage

```
OpEC4photo(obsDat, iVcmax = 60, iVpmax = 120, iVpr = 80, iJmax = 400,
  co2 = 380, o2 = 210, level = 0.95)
```

Arguments

obsDat	observed assimilation data, which should be a data frame or matrix. The first column should be observed net assimilation rate (μ mol m^{-2} s^{-1}). The second column should be the observed quantum flux (μ mol m^{-2} s^{-1}). The third column should be observed temperature of the leaf (Celsius). The fourth column should be the observed relative humidity in proportion (e.g. 0.7).
iVcmax	initial value for Vcmax (default = 60).
iVpmax	initial value for Vpmax (default = 120).
iVpr	initial value for Vpr (default = 80).
iJmax	initial value for Jmax (default = 400).
co2	atmospheric CO2 concentration (ppm), default = 380.
o2	atmospheric O2 concentration (mmol/mol), default = 210.
level	level for the confidence intervals.

Value

an object of class `OpEC4photo`. Notice that these are the new-style S4 classes.

Examples

```
data(obsNaid)
## These data are from Naidu et al. (2003)
## in the correct format
res <- OpEC4photo(obsNaid)
## Other example using Beale, Bint and Long (1996)
data(obsBea)
resB <- OpEC4photo(obsBea)
```

OpMaizeGro *Optimization of dry biomass partitioning coefficients.*

Description

Optimizes dry biomass partitioning coefficients using constrained optimization (see below).

Usage

```
OpMaizeGro(phen = 1, iCoef = NULL, cTT, WetDat, data, plant.day = NULL,
  emerge.day = NULL, harvest.day = NULL, plant.density = 7,
  timestep = 1, lat = 40, canopyControl = list(),
  MaizeSeneControl = list(), photoControl = list(),
  MaizePhenoControl = list(), MaizeCAllocControl = list(),
  laiControl = list(), soilControl = list(), MaizeNitroControl = list(),
  centuryControl = list(), op.method = c("optim", "nlminb"),
  verbose = FALSE, ...)
```

Arguments

phen	integer taking values 1 through 6 which indicate the phenological stage being optimized. If all of the phenological stages need to be optimized then use zero (0).
iCoef	initial vector of size 24 for the dry biomass partitioning coefficients.
cTT	
WetDat	Weather data.
data	observed data.
plant.day	
emerge.day	
harvest.day	
plant.density	
canopyControl	see MaizeGro
seneControl	see MaizeGro
photoControl	see MaizeGro
phenoControl	see MaizeGro
soilControl	see MaizeGro
nitroControl	see MaizeGro
centuryControl	see MaizeGro
op.method	Optimization method. Whether to use optim or nlminb
verbose	Displays additional information, originally used for debugging.
...	additional arguments passed to optim or constrOptim.

Details

The optimization is done over the MaizeGro function. The OpMaizeGro function is a wrapper for optim and the constrOpBioGro is a wrapper for constrOptim.

---

plot.BioGro *Plotting function for BioGro objects*

Description

By default it plots stem, leaf, root, rhizome and LAI for a BioGro object. Optionally, the observed data can be plotted.

Usage

```
## S3 method for class 'BioGro'
plot(x, obs = NULL, stem = TRUE, leaf = TRUE,
  root = TRUE, rhizome = TRUE, LAI = TRUE, grain = TRUE, xlab = NULL,
  ylab = NULL, pch = 21, lty = 1, lwd = 1, col = c("blue", "green",
  "red", "magenta", "black", "purple"), x1 = 0.1, y1 = 0.8,
  plot.kind = c("DB", "SW"), ...)
```

Arguments

x	BioGro object.
obs	optional observed data object (format following the OpBioGro function .
stem	whether to plot simulated stem (default = TRUE).
leaf	whether to plot simulated leaf (default = TRUE).
root	whether to plot simulated root (default = TRUE).
rhizome	whether to plot simulated rhizome (default = TRUE).
grain	whether to plot simulated grain (default = TRUE).
LAI	whether to plot simulated LAI (default = TRUE).
pch	point character.
lty	line type.
lwd	line width.
col	Control of colors.
x1	position of the legend. x coordinate (0-1).
y1	position of the legend. y coordinate (0-1).
plot.kind	DB plots dry biomass and SW plots soil water.
...	Optional arguments.

Details

This function uses internally xyplot in the 'lattice' package.

See Also

BioGro OpBioGro

`plot.MaizeGro` *Plotting function for MaizeGro objects*

Description

By default it plots stem, leaf, root, rhizome and LAI for a `MaizeGro` object. Optionally, the observed data can be plotted.

Usage

```
## S3 method for class 'MaizeGro'
plot(x, obs = NULL, stem = TRUE, leaf = TRUE,
  root = TRUE, LAI = TRUE, grain = TRUE, xlab = NULL, ylab = NULL,
  pch = 21, lty = 1, lwd = 1, col = c("blue", "green", "red", "black",
  "purple"), x1 = 0.1, y1 = 0.8, plot.kind = c("DB", "SW", "LAI",
  "pheno"), ...)
```

Arguments

`x`	`MaizeGro` object.
`obs`	optional observed data object (format following the `OpMaizeGro` function .
`stem`	whether to plot simulated stem (default = TRUE).
`leaf`	whether to plot simulated leaf (default = TRUE).
`root`	whether to plot simulated root (default = TRUE).
`rhizome`	whether to plot simulated rhizome (default = TRUE).
`grain`	whether to plot simulated grain (default = TRUE).
`LAI`	whether to plot simulated LAI (default = TRUE).
`pch`	point character.
`lty`	line type.
`lwd`	line width.
`col`	Control of colors.
`x1`	position of the legend. x coordinate (0-1).
`y1`	position of the legend. y coordinate (0-1).
`plot.kind`	DB plots dry biomass and SW plots soil water.
`...`	Optional arguments.

Details

This function uses internally `xyplot` in the 'lattice' package.

See Also

`MaizeGro` `OpMaizeGro`

`plot.MCMCBioGro` *Plotting function fo the MCMCBioGro class*

Description

Powerful plotting function to make a variety of plots regarding the `MCMCBioGro` class (output). It plots the residual sum of square progression, observed vs. fitted, residuals vs. fitted, trace of the coefficients and density.

Usage

```
## S3 method for class 'MCMCBioGro'
plot(x, x2 = NULL, x3 = NULL, plot.kind = c("rss",
  "OF", "RF", "OFT", "trace", "density"), type = c("l", "p"), coef = 1,
  cols = c("blue", "green", "red", "magenta", "black", "purple"), ...)
```

Arguments

`x`	Object of class `MCMCBioGro`
`x2`	Optional object of class `MCMCBioGro`
`x3`	Optional object of class `MCMCBioGro`
`plot.kind`	Kind of plot. See details.
`type`	Point of line as in `xyplot`
`coef`	Whether to plot dry biomass partitioning coeficients (2) or Vmax and alpha (1).
`cols`	Colors. Modify if they don't suit you.
`...`	Additional arguments passed to the underlying `xyplot` function. Some can be really useful. See details.

Details

Kind of plots that can be produced

rss: Residual Sum of Squares progression. OF: Observed vs. Fitted RF: Residual vs. Fitted OFT: Observed and Fitted with thermal time as the x-axis. trace: trace for the parameters density: density for the parameters

biomass used. If coef=1 then Vmax instead. At this point it is highly and alpha are not optimized at this stage so shouldn't be plotted either (it will be a straight line if not

To choosing a subset of the 24 dry biomass partitioning coefficients use the subset option as you would in the xyplot using nams as the name. For example, `plot(x,plot.kind="trace",subset=nams=="kLeaf_`
will select it for the leaf at the first phenological stage.

Value

A `lattice` plot.

See Also

MCMCBioGro

Examples

```
## See the MCMCBioGro function
```

---

plot.MCMCc4photo *Plotting function for MCMCc4photo objects*

Description

By default it prints the trace of the four parameters being estimated by the MCMCc4photo function. As an option the density can be plotted.

Usage

```
## S3 method for class 'MCMCc4photo'
plot(x, x2 = NULL, x3 = NULL, plot.kind = c("trace",
  "density"), type = c("l", "p"), burnin = 1, cols = c("blue", "green",
  "purple"), prior = FALSE, pcol = "black", ...)
```

Arguments

x	MCMCc4photo object.
x2	optional additional MCMCc4photo object.
x3	optional additional MCMCc4photo object.
plot.kind	'trace' plots the iteration history and 'density' plots the kernel density.
type	only the options for line and point are offered.
burnin	this will remove part of the chain that can be considered burn-in period. The plots will no include this part.
cols	Argument to pass colors to the line or points being plotted.
prior	Whether to plot the prior density. It only works when x2 = NULL and x3 = NULL. Default is FALSE.
pcol	Color used for plotting the prior density.
...	Optional arguments.

Details

This function uses internally xyplot, densityplot and panel.mathdensity both in the 'lattice' package.

See Also

MCMCc4photo

plot.MCMCEc4photo *Plottin function for MCMCEc4photo objects*

Description

By default it prints the trace of the four parameters being estimated by the `MCMCEc4photo` function. As an option the density can be plotted.

Usage

```
## S3 method for class 'MCMCEc4photo'
plot(x, x2 = NULL, x3 = NULL, type = c("trace",
  "density"), ...)
```

Arguments

`x`	`MCMCEc4photo` object.
`x2`	optional additional `link{MCMCEc4photo}` object.
`x3`	optional additional `link{MCMCEc4photo}` object.
`type`	'trace' plots the iteration history and 'density' plots the kernel density.
`...`	Optional arguments.

Details

This function uses internally `xyplot` and `densityplot` both in the 'lattice' package.

See Also

`MCMCEc4photo`

plot.mOpc3photo *Plotting method*

Description

Plotting method

Usage

```
## S3 method for class 'mOpc3photo'
plot(x, parm = c("vcmax", "jmax"), ...)
```

plot.mOpc4photo *Plotting method*

Description

Plotting method

Usage

```
## S3 method for class 'mOpc4photo'
plot(x, parm = c("vmax", "alpha"), ...)
```

plot.willowGro *Plotting function for willowGro objects*

Description

By default it plots stem, leaf, root, rhizome and LAI for a `willowGro` object. Optionally, the observed data can be plotted.

Usage

```
## S3 method for class 'willowGro'
plot(x, obs = NULL, stem = TRUE, leaf = TRUE,
  root = TRUE, rhizome = TRUE, LAI = TRUE, grain = TRUE, xlab = NULL,
  ylab = NULL, pch = 21, lty = 1, lwd = 1, col = c("blue", "green",
  "red", "magenta", "black", "purple"), x1 = 0.1, y1 = 0.8,
  plot.kind = c("DB", "SW"), ...)
```

Arguments

`x`	`willowGro` object.
`obs`	optional observed data object (format following the `OpwillowGro` function .
`stem`	whether to plot simulated stem (default = TRUE).
`leaf`	whether to plot simulated leaf (default = TRUE).
`root`	whether to plot simulated root (default = TRUE).
`rhizome`	whether to plot simulated rhizome (default = TRUE).
`grain`	whether to plot simulated grain (default = TRUE).
`LAI`	whether to plot simulated LAI (default = TRUE).
`pch`	point character.
`lty`	line type.
`lwd`	line width.

col	Control of colors.
x1	position of the legend. x coordinate (0-1).
y1	position of the legend. y coordinate (0-1).
plot.kind	DB plots dry biomass and SW plots soil water.
...	Optional arguments.

Details

This function uses internally xyplot in the 'lattice' package.

See Also

willowGro OpwillowGro

---

plotAC *plot A/Ci curve*

Description

Function to plot A/Ci curves

Usage

```
plotAC(data, fittd, id.col = 1, trt.col = 2, ylab = "CO2 Uptake",
  xlab = "Ci", by = c("trt", "ID"), type = c("p", "smooth"))
```

Arguments

data	Input data in the format needed for the mOpc4photo; assumed to have the following structure col 1: trt col 2 (optional): other treatment factor col 2: Assimilation col 3: Quantum flux col 4: Temperature col 5: Relative humidity col 6: Intercellular CO2 col 7: Reference CO2
fittd	Optional fitted values.
id.col	Specify which column has the ids. Default is col 1.
trt.col	Specify which column has the treatments. Default is col 2. If no treatment is specified then use 1.
ylab	Label for the y-axis.
xlab	Label for the x-axis.
by	Whether to plot by id or by treatment.
type	this argument is passed to the xyplot. It changes the plotting symbols behavior.

Details

A small helper function that can be used to easily plot multiple A/Ci curves

Value

NULL, creates plot

Author(s)

Fernando E. Miguez

See Also

See Also `xyplot`.

Examples

```
data(aci)
plotAC(aci, trt.col=1)
```

---

plotAQ *plot A/Q curve*

Description

Function to plot A/Q curves

Usage

```
plotAQ(data, fittd, id.col = 1, trt.col = 2, ylab = "CO2 Uptake",
  xlab = "Quantum flux", by = c("trt", "ID"), type = "o", ...)
```

Arguments

`data`	is assumed to have the following structure col 1: trt col 2 (optional): other treatment factor col 2: Assimilation col 3: Quantum flux col 4: Temperature col 5: Relative humidity col 6 (optional): Reference CO2
`fittd`	
`id.col`	
`trt.col`	
`ylab`	
`xlab`	
`by`	
`type`	
`...`	

Value

NULL, creates plot

Author(s)

Fernando E. Miguez

`predict.Opc3photo` *Predict method*

Description

Predict method

Usage

```
## S3 method for class 'Opc3photo'
predict(object, newdata, ...)
```

`predict.Opc4photo` *Predict method*

Description

Predict method

Usage

```
## S3 method for class 'Opc4photo'
predict(object, newdata, ...)
```

`print.BioGro` *printing method for BioGro*

Description

printing method for BioGro

Usage

```
## S3 method for class 'BioGro'
print(x, level = 1, ...)
```

Arguments

`x`

print.MaizeGro *printing method for MaizeGro*

Description

printing method for MaizeGro

Usage

```
## S3 method for class 'MaizeGro'
print(x, level = 1, ...)
```

Arguments

x

print.MCMCBioGro *printing method for MCMCBioGro*

Description

printing method for MCMCBioGro

Usage

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

Arguments

x

print.MCMCc4photo *Function for printing the MCMCc4photo objects*

Description

Function for printing the MCMCc4photo objects

Usage

```
## S3 method for class 'MCMCc4photo'
print(x, burnin = 1, level = 0.95, digits = 1, ...)
```

print.MCMCEc4photo *Print an MCMCEc4photo object*

Description

This functions doesn't just print the object components, but it also computes quantiles according to the `level` argument below and a correlation matrix as well as a summary for each parameter.

Usage

```
## S3 method for class 'MCMCEc4photo'
print(x, level = 0.95, ...)
```

Arguments

`x`	`MCMCEc4photo` object
`level`	specified `level` for the Highest Posterior Density region.
`...`	Optional arguments

Details

The Highest Posterior Density region is calculated using the `quantile` function. The correlation matrix is computed using the `cor` function. The summaries for each parameter are computed using the `summary` function.

See Also

`MCMCEc4photo`

print.mOpc3photo *Printing method*

Description

Printing method

Usage

```
## S3 method for class 'mOpc3photo'
print(x, ...)
```

print.mOpc4photo *Printing method*

Description

Printing method

Usage

```
## S3 method for class 'mOpc4photo'
print(x, ...)
```

print.OpBioGro *Print an OpBioGro object*

Description

Print an object generated by OpBioGro

Usage

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

Arguments

x	Object returned from BioGro optimization (Collatz model).
...	Optional arguments.

See Also

OpBioGro

print.Opc3photo *Display methods for Opc4photo and OpEC4photo*

Description

Display methods for Opc4photo and OpEC4photo

Usage

```
## S3 method for class 'Opc3photo'
print(x, digits = 2, ...)
```

print.Opc4photo *Display methods for Opc4photo and OpEC4photo*

Description

Display methods for Opc4photo and OpEC4photo

Usage

```
## S3 method for class 'Opc4photo'
print(x, digits = 2, ...)
```

print.OpEC4photo *Print an OpEC4photo object*

Description

Print an object generated by OpEC4photo

Usage

```
## S3 method for class 'OpEC4photo'
print(x, ...)
```

Arguments

x	Object returned from C4 photosynthesis optimization (von Caemmerer model).
...	Optional arguments.

See Also

OpEC4photo

`Rmiscanmod` *RUE-based model to calculate miscanthus growth and yield.*

Description

Simple model to calculate crop growth and yield based on MISCANMOD (see references).

Usage

```
Rmiscanmod(data, RUE = 2.4, LER = 0.01, Tb = 10, k = 0.67,
  LAIdrd = 0.8, LAIStop = 1.8, RUEdrd = 1.3, RUEStop = 2.5,
  SMDdrd = -30, SMDStop = -120, FieldC = 45, iWatCont = 45,
  a = 6682.2, b = -0.33, soildepth = 0.6)
```

Arguments

`data`	data.frame or matrix described in details.
`RUE`	Radiation use efficiency (g/MJ).
`LER`	Leaf expansion rate LAI/GDD.
`Tb`	Base Temperature (Celsius).
`k`	extinction coefficient of light in the canopy.
`LAIdrd`	Leaf Area Index 'down regulation decline'.
`LAIStop`	Leaf Area Index 'down regulation decline' threshold .
`RUEdrd`	Radiation Use Efficieny 'down regulation decline'.
`RUEStop`	Radiation Use Efficieny 'down regulation decline' threshold.
`SMDdrd`	Soil Moisture Deficit 'down regulation decline'.
`SMDStop`	Soil Moisture Deficit 'down regulation decline' threshold.
`FieldC`	Soil field capacity.
`iWatCont`	Initial water content.
`a`	Soil parameter.
`b`	Soil parameter.
`soildepth`	Soil depth.

Details

The data.frame or matrix should contain

column 1: year column 2: month column 3: day column 4: JD column 5: max T (Celsius) column 6: min T (Celsius) column 7: PPFD or solar radiation divided by 2 (MJ/m2) column 8: Potential evaporation column 9: precip (mm)

Value

returns a list

References

Clifton-Brown, J. C.; Neilson, B.; Lewandowski, I. and Jones, M. B. The modelled productivity of Miscanthus x giganteus (GREEF et DEU) in Ireland. Industrial Crops and Products, 2000, 12, 97-109.

Clifton-brown, J. C.; Stampfl, P. F. and Jones, M. B. Miscanthus biomass production for energy in Europe and its potential contribution to decreasing fossil fuel carbon emissions. Global Change Biology, 2004, 10, 509-518.

Examples

```
## Need to get an example data set and then run it
## Not run:
data(WD1979)

res <- Rmiscanmod(WD1979)

## convert to Mg/ha

Yld <- res$Yield / 100

xyplot(Yld ~ 1:365 ,
       xlab='doy',
       ylab='Dry biomass (Mg/ha)')

## although the default value for Field Capacity is 45
## a more reasonable value is closer to 27



## End(Not run)
```

`rootDist` *Finds rootDist for the soilML function*

Description

This function returns a value for rootDist based on the arguments layers, rootDepth, depthsp, and rfl. This value then factors into several equations in the primary function soilML.

Usage

```
rootDist(layers, rootDepth, depthsp, rfl)
```

Arguments

`layers`	Integer used to specify number of soil layers, should be greater than 2
`rootDepth`	depth of the root in soil
`depthsp`	represents vaule of the sequence depths in soilML

rfl	Root factor lambda. A Poisson distribution is used to simulate the distribution of roots in the soil profile and this parameter can be used to change the lambda parameter of the Poisson.

RsqC4photo *R-squared for C4 photosynthesis simulation (Collatz model)*

Description

This is an auxiliary function which is made available in case it is useful. It calculates the R-squared based on observed assimilation (or stomatal conductance) data and coefficients for the Collatz C4 photosynthesis model. The only coefficients being considered are Vcmax and alpha as described in the Collatz paper. At the moment it does not optimize k; this will be added soon. Notice that to be able to optimize k A/Ci type data are needed.

Usage

```
RsqC4photo(data, vmax = 39, alph = 0.04, kparm = 0.7, theta = 0.83,
  beta = 0.93, Rd = 0.8, iupperT = 37.5, ilowerT = 3, Catm = 380,
  b0 = 0.08, b1 = 3, StomWS = 1, response = c("Assim", "StomCond"))
```

Arguments

data	observed assimilation data, which should be a data frame or matrix. The first column should be observed net assimilation rate (μ mol m^{-2} s^{-1}). The second column should be the observed quantum flux (μ mol m^{-2} s^{-1}). The third column should be observed temperature of the leaf (Celsius). The fourth column should be the observed relative humidity in proportion (e.g. 0.7).
vmax	Vcmax (default = 39); for more details see the c4photo function.
alph	alpha as in Collatz (default = 0.04); for more details see the c4photo function.
kparm	k as in Collatz (default = 0.70); for more details see the c4photo function.
theta	theta as in Collatz (default = 0.83); for more details see the c4photo function.
beta	beta as in Collatz (default = 0.93); for more details see the c4photo function.
Rd	Rd as in Collatz (default = 0.8); for more details see the c4photo function.
StomWS	StomWS as in Collatz (default = 1); for more details see the c4photo function.
Catm	Atmospheric CO2 in ppm (or μmol/mol).
b0	Intercept for the Ball-Berry model.
b1	Slope for the Ball-Berry model.
response	Use 'Assim' if you want an R^2 for assimilation data and use 'StomCond' if you want an R^2 for stomatal conductance data.

Value

a numeric object

It simply returns the R^2 value for the given data and coefficients.

Examples

```
data(obsNaid)
## These data are from Naidu et al. (2003)
## in the correct format
res <- RsqC4photo(obsNaid)
## Other example using Beale, Bint and Long (1996)
data(obsBea)
resB <- RsqC4photo(obsBea)
```

RsqEC4photo *R-squared for C4 photosynthesis simulation (von Caemmerer model)*

Description

This is an auxiliary function which is made available in case it is useful. It calculates the R-squared based on observed assimilation (or stomatal conductance) data and coefficients for the von Caemmerer C4 photosynthesis model. The only coefficients being considered are Vcmax, Vpmax, Vpr and Jmax.

Usage

```
RsqEC4photo(obsDat, iVcmax = 60, iVpmax = 120, iVpr = 80, iJmax = 400,
  co2 = 380, o2 = 210, type = c("Assim", "StomCond"))
```

Arguments

`obsDat`	observed assimilation data, which should be a data frame or matrix. The first column should be observed net assimilation rate (μ mol m^{-2} s^{-1}). The second column should be the observed quantum flux (μ mol m^{-2} s^{-1}). The third column should be observed temperature of the leaf (Celsius). The fourth column should be the observed relative humidity in proportion (e.g. 0.7).
`iVcmax`	Maximum rubisco activity (μ mol m^{-2} s^{-1}).
`iVpmax`	Maximum PEP carboxylase activity (μ mol m^{-2} s^{-1}).
`iVpr`	PEP regeneration rate (μ mol m^{-2} s^{-1}).
`iJmax`	Maximal electron transport rate (μmol electrons m^{-2} s^{-1}).
`co2`	atmospheric carbon dioxide concentration (ppm or μbar) (default = 380).
`o2`	atmospheric oxygen concentration (mbar) (default = 210).
`type`	Use `"Assim"` if you want an R^2 for assimilation data and use `"StomCond"` if you want an R^2 for stomatal conductance data.

Value

a `numeric` object

It simply returns the R^2 value for the given data and coefficients.

Examples

```
data(obsNaid)
obs <- obsNaid
## These data are from Naidu et al. (2003)
## in the correct format
res <- RsqEC4photo(obs)
## Other example using Beale, Bint and Long (1996)
data(obsBea)
obsD <- obsBea
resB <- RsqEC4photo(obsD)
```

`RssBioGro` *Residual sum of squares for BioGro.*

Description

Computes residual sum of squares for the `BioGro` function.

Usage

```
RssBioGro(obs, sim)
```

Arguments

`obs`	Observed data.
`sim`	Simulated data.

Value

Atomic vector with the residual sum of squares.

Author(s)

Fernando E. Miguez

See Also

See Also `BioGro`.

Examples

```
## A simple example
data(annualDB)
data(EngWea94i)
res <- BioGro(EngWea94i)
RssBioGro(annualDB,res)
```

`RssMaizeGro` *Very simple function to compare the distance between simulated and observed data for the BioGro function Need to add an argument such as pc.sigmas "plant component sigmas" If variability of the plant component is known*

Description

Very simple function to compare the distance between simulated and observed data for the BioGro function Need to add an argument such as pc.sigmas "plant component sigmas" If variability of the plant component is known

Usage

```
RssMaizeGro(obs, sim)
```

`RUEmod` *Radiation use efficiency based model*

Description

Simulates leaf area index, biomass and light interception. Based on the Monteith (1973) equations, adapted for Miscanthus by Clifton Brown et al.(2001) (see references).

Usage

```
RUEmod(Rad, T.a, doy.s = 91, doy.f = 227, lai.c = 0.0102, rue = 2.4,
  T.b = 10, k = 0.68)
```

Arguments

`Rad`	Daily solar radiation (MJ ha^{-2}).
`T.a`	Daily average temperature (Fahrenheit).
`doy.s`	first day of the growing season, default 91.
`doy.f`	last day of the growing season, default 227.
`lai.c`	linear relationship between growing degree days and leaf area index.
`rue`	radiation use efficiency, default 2.4.
`T.b`	base temperature for calculating growing degree days, default 10.
`k`	light extinction coefficient, default 0.68.

Value

a `list` structure with components

`doy`	day of the year.
`lai.cum`	cumulative leaf area index.
`AG.cum`	cumulative above ground dry biomass (Mg ha^{-1}).
`AGDD`	cumulative growing degree days.
`Int.e`	Intercepted solar radiation.

Note

This empirical model is useful but it has limitations.

References

Monteith (1973) *Principles of Environmental Physics*. Edward Arnold, London. UK.

Clifton-Brown, J.C., Long, S.P. and Jorgensen, U. with contributions from S.A. Humphries, Schwarz, K.-U. and Schwarz, H. (2001) *Miscanthus Productivity*. Ch 4. In: Miscanthus for Energy and Fibre. Edited by: Jones, Michael B. and Walsh, Mary. James & James (Science Publishers), London, UK.

See Also

`RUEmodMY`

Examples

```
## See RUEmodMY
```

`RUEmodMY` *Radiation use efficiency based model*

Description

Same as `RUEmod` but it handles multiple years.

Usage

```
RUEmodMY(weatherdatafile, doy.s = 91, doy.f = 227, ...)
```

Arguments

`weatherdatafile`	weather data file (see example).
`doy.s`	first day of the growing season, default 91.
`doy.f`	last day of the growing season, default 227.
`...`	additional arguments to be passed to the `RUEmod` function.

Value

a `data.frame` structure with components

Examples

```
## weather data from Champaign, IL
data(cmiWet)
tmp1 <- RUEmodMY(cmiWet)

xyplot(AG.cum ~ doy | factor(year), type='l', data = tmp1,
       lwd=2,
       ylab=expression(paste('dry biomass (Mg ',ha^-1,')')),
       xlab='DOY')
```

---

showSoilType *the function that deinfes the soil types*

Description

the function that deinfes the soil types

Usage

```
showSoilType(soiltype)
```

Arguments

`soiltype`	an integer from 0 to 10 that coresopnds to a type of soil

---

simDat2 *Simulated biomass data.*

Description

Simulated data produced by `BioGro`.

Format

A data frame with 5 observations on the following 6 variables.

list('TT') a numeric vector

list('Stem') a numeric vector

list('Leaf') a numeric vector

list('Root') a numeric vector

list('Rhiz') a numeric vector

list('LAI') a numeric vector

Details

~~ If necessary, more details than the description above ~~

Source

~~ reference to a publication or URL from which the data were obtained ~~

References

~~ possibly secondary sources and usages ~~

Examples

```
data(simDat2)
## maybe str(simDat2) ; plot(simDat2) ...
```

SoilEvapo *Soil Evaporation*

Description

Calculates soil evaporation

Usage

```
SoilEvapo(LAI, k, AirTemp, IRad, awc, FieldC, WiltP, winds, RelH)
```

Arguments

LAI	Leaf Area Index.
k	~~Describe k here~~
AirTemp	Air temperature.
IRad	Incident radiation.
awc	Available water content.
FieldC	Field capacity.
WiltP	Wilting point.
winds	Wind speed.
RelH	Relative humidty.

Details

The style of the code is C like because this is a prototype for the underlying C (like so many other functions in this package). I leave it here for future development.

Value

Returns a single value of soil Evaporation in Mg H20 per hectare.

Author(s)

Fernando Miguez

See Also

Source code :)

Examples

```
SoilEvapo(LAI=3,k=0.68,AirTemp=20,IRad=1000,awc=0.3,FieldC=0.4,WiltP=0.2,winds=3,RelH=0.8)
```

---

soilML *soil multi-layered*

Description

Simulates soil water content for a layered soil.

Usage

```
soilML(precipt, CanopyT, cws, soilDepth, FieldC, WiltP, phi1 = 0.01,
  phi2 = 10, wsFun = c("linear", "logistic", "exp", "none"), rootDB,
  soilLayers = 3, LAI, k, AirTemp, IRad, winds, RelH, soilType = 10,
  hydrDist = 0, rfl = 0.3)
```

Arguments

precipt	Precipitation (mm).
CanopyT	Canopy transpiration.
cws	Current water status. Vector of length equal to soilLayers.
soilDepth	Rooting depth.
FieldC	Field capacity.
WiltP	Wilting point.
phi1	See wtrstr.
phi2	See wtrstr.
wsFun	See wtrstr.
rootDB	Root biomass (Mg/ha).
soilLayers	Integer used to specify the number of soil layers.
LAI	Leaf area index.
k	Light extinction coefficient.

AirTemp	Air temperature (Celsius).
IRad	Direct irradiance ($\mu\ m^-2\ s^-1$).
winds	Wind speed (m/s).
RelH	Relative humidity (0-1).
soilType	See showSoilType.
hydrDist	Zero or otherwise positive integer. Zero does not calculate hydraulic distribution, otherwise does.
rfl	Root factor lambda. A Poisson distribution is used to simulate the distribution of roots in the soil profile and this parameter can be used to change the lambda parameter of the Poisson.

Value

optiontocalculaterootdept

rootfrontvelocity

dap

matrix with 8 (if hydrDist=0) or 12 (if hydrDist > 0).

Author(s)

Fernando E. Miguez

See Also

See Also wtrstr.

Examples

```
layers <- 5
ans <- soilML(precipt=2, CanopyT=2, cws = rep(0.3,layers),
soilDepth=1.5, FieldC=0.33, WiltP=0.13, rootDB=2, soilLayers=layers,
LAI=3, k=0.68, AirTemp=25,IRad=500, winds=2, RelH=0.8, soilType=6,
hydrDist=1)
ans
```

SoilType *takes an interger from 0 to 10 that coresponds to a specifically defined soil type and returns the composition of the soil in a list.*

Description

takes an interger from 0 to 10 that coresponds to a specifically defined soil type and returns the composition of the soil in a list.

Usage

```
SoilType(soiltype)
```

Arguments

soiltype	an integer from 0 to 10 that coresopnds to a type of soil

---

summary.OpBioGro *This function will implement simple calculations of predicted and residuals for the OpBioGro function*

Description

This function will implement simple calculations of predicted and residuals for the OpBioGro function

Usage

```
## S3 method for class 'OpBioGro'
summary(object, ...)
```

Arguments

object	

---

summary.Opc4photo *This function will implement simple calculations of predicted and residuals for the Opc4photo function*

Description

This function will implement simple calculations of predicted and residuals for the Opc4photo function

Usage

```
## S3 method for class 'Opc4photo'
summary(object, ...)
```

summary.OpEC4photo *This function will implement simple calculations of predicted and residuals for the OpEC4photo function*

Description

This function will implement simple calculations of predicted and residuals for the OpEC4photo function

Usage

```
## S3 method for class 'OpEC4photo'
summary(object, ...)
```

sunML *Sunlit shaded multi-layer model*

Description

Simulates the light microenvironment in the canopy based on the sunlit-shade model and the multiple layers.

Usage

```
sunML(Idir, Idiff, LAI = 8, nlayers = 8, cos.theta = 0.5, kd = 0.7,
  chi.l = 1, heightf = 3)
```

Arguments

I.dir	direct light (quantum flux), ($\mu mol\ m^{-2}\ s^{-1}$).
I.diff	indirect light (diffuse), ($\mu mol\ m^{-2}\ s^{-1}$).
LAI	leaf area index, default 8.
nlayers	number of layers in which the canopy is partitioned, default 8.
kd	extinction coefficient for diffuse light.
chi.l	The ratio of horizontal:vertical projected area of leaves in the canopy segment.
cos.theta	cosine of θ, solar zenith angle.

Value

a list structure with components

Vectors size equal to the number of layers.

Examples

```
## Not run:
res2 <- sunML(1500,200,3,10)

xyplot(Fsun + Fshade ~ c(1:10), data=res2,
       ylab='LAI',
       xlab='layer',
       type='l',lwd=2,col=c('blue','green'),
       lty=c(1,2),
       key=list(text=list(c('Direct','Diffuse')),lty=c(1,2),
                cex=1.2,lwd=2,lines=TRUE,x=0.7,y=0.5,
                col=c('blue','green')))

## End(Not run)
```

TempToDdryA *Returns a value for TempToDdryA*

Description

Takes a value for Temp as defined by the SoilEvapo function and returns a value for DdryA which factors into variable PsychParam which in turn helps define the Evaporation.

Usage

```
TempToDdryA(Temp)
```

Arguments

Temp	Temperature

TempToLHV *Returns a value for TempToLHV*

Description

Takes a value for Temp as defined by the SoilEvapo function and returns a value for LHV which factors into variable PsychParam which in turn helps define the Evaporation.

Usage

```
TempToLHV(Temp)
```

Arguments

Temp	Temperature

`TempToSFS` *Returns a value for SlopeFS from a function of temperature*

Description

Takes a value for Temp as defined by the SoilEvapo function and returns a value for SlopeFS which helps define the Evaporation.

Usage

```
TempToSFS(Temp)
```

Arguments

`Temp`	Temperature

`TempToSWVC` *Returns a value for TempToSWC*

Description

Takes a value for Temp as defined by the SoilEvapo function and returns a value for SWVC which factors into variable DeltaPVa which in turn helps define the Evaporation.

Usage

```
TempToSWVC(Temp)
```

Arguments

`Temp`	Temperature

valid_dbp *Validate dry biomass partitioning coefficients*

Description

It attempts to check the requirements of the dry biomass partitioning coefficients.

Usage

```
valid_dbp(x, tol = 0.001)
```

Arguments

x	Vector of length equal to 25 containing the dry biomass partitioning coefficients for the 6 phenological stages as for example produced by phenoParms.
tol	Numerical tolerance passed to the all.equal function.

Value

It will return the vector of coefficients unchanged if no errors are detected.

Author(s)

Fernando E. Miguez

See Also

BioGro

Examples

```
xx <- as.vector(unlist(phenoParms())[7:31])
valid_dbp(xx)
```

WD1979 *Randomly picked dataset from the Illinois area from 1979*

Description

Data from the Illinois area from one point from the 32km grid from NOAA from 1979. the purpuse is to illustrate the Rmiscanmod function.

Format

A data frame with 365 observations on the following 9 variables.

list('year') year

list('month') month (not really needed)

list('day') day of the month (not really needed)

list('JD') day of the year (1-365)

list('maxTemp') maximum temperature (Celsius)

list('minTemp') minimum temperature (Celsius)

list('SolarR') solar radiation (MJ/m2)

list('PotEv') potential evaporation (kg/m2). Approx. mm.

list('precip') precipitation (kg/m2). Approx. mm.

Source

`http://www.noaa.gov/`

Examples

```
data(WD1979)
summary(WD1979)
```

---

weach *Simulates the hourly conditions from daily*

Description

Manipulates weather data in the format obtained from WARM (see link below) and returns the format and units needed for most functions in this package. This function should be used for one year at a time. It returns hourly (or sub-daily) weather information.

Usage

```
weach(X, lati, ts = 1, temp.units = c("Fahrenheit", "Celsius"),
  rh.units = c("percent", "fraction"), ws.units = c("mph", "mps"),
  pp.units = c("in", "mm"), ...)
```

Arguments

X	a matrix (or data frame) containing weather information. The input format is strict but it is meant to be used with the data usually obtained from weather stations in Illinois. The data frame should have 11 columns (see details).
lati	latitude at the specific location
ts	timestep for the simulation of sub-daily data from daily. For example a value of 3 would return data every 3 hours. Only divisors of 24 work (i.e. 1,2,3,4, etc.).

temp.units	Option to specify the units in which the temperature is entered. Default is Farenheit.
rh.units	Option to specify the units in which the relative humidity is entered. Default is percent.
ws.units	Option to specify the units in which the wind speed is entered. Default is miles per hour.
pp.units	Option to specify the units in which the precipitation is entered. Default is inches.
list()	additional arguments to be passed to lightME

Details

This function was originally used to transform daily data to hourly data. Some flexibility has been added so that other units can be used. The input data used originally looked as follows.

\itemcol 1year \itemcol 2day of the year (1–365). Does not consider leap years. \itemcol 3total daily solar radiation (MJ/m^2). \itemcol 4maximum temperature (Fahrenheit). \itemcol 5minimum temperature (Fahrenheit). \itemcol 6average temperature (Fahrenheit). \itemcol 7maximum relative humidity (%). \itemcol 8minimum relative humidity (%). \itemcol 9average relative humidity (%). \itemcol 10average wind speed (miles per hour). \itemcol 11precipitation (inches).

All the units above are the defaults but they can be changed as part of the arguments.

Value

a matrix returning hourly (or sub-daily) weather data. Dimensions 8760 (if hourly) by 8.

Examples

```
data(cmi0506)
tmp1 <- cmi0506[cmi0506$year == 2005,]
wet05 <- weach(tmp1,40)

# Return data every 3 hours
wet05.3 <- weach(tmp1,40,ts=3)
```

---

weach365 *if the function CheckLeapYear returns 0 this function will be used*

Description

if the function CheckLeapYear returns 0 this function will be used

Usage

```
weach365(X, lati, ts = 1, temp.units = c("Farenheit", "Celsius"),
  rh.units = c("percent", "fraction"), ws.units = c("mph", "mps"),
  pp.units = c("in", "mm"), ...)
```

Arguments

X	a matrix (or data frame) containing weather information. The input format is strict but it is meant to be used with the data usually obtained from weather stations in Illinois. The data frame should have 11 columns (see details).
lati	latitude at the specific location
ts	timestep for the simulation of sub-daily data from daily. For example a value of 3 would return data every 3 hours. Only divisors of 24 work (i.e. 1,2,3,4, etc.).
temp.units	Option to specify the units in which the temperature is entered. Default is Farenheit.
rh.units	Option to specify the units in which the relative humidity is entered. Default is percent.
ws.units	Option to specify the units in which the wind speed is entered. Default is miles per hour.
pp.units	Option to specify the units in which the precipitation is entered. Default is inches.
list()	additional arguments to be passed to lightME

---

weach366 *if the function CheckLeapYear returns 1 this function will be used*

Description

if the function CheckLeapYear returns 1 this function will be used

Usage

```
weach366(X, lati, ts = 1, temp.units = c("Farenheit", "Celsius"),
  rh.units = c("percent", "fraction"), ws.units = c("mph", "mps"),
  pp.units = c("in", "mm"), ...)
```

Arguments

X	a matrix (or data frame) containing weather information. The input format is strict but it is meant to be used with the data usually obtained from weather stations in Illinois. The data frame should have 11 columns (see details).
lati	latitude at the specific location
ts	timestep for the simulation of sub-daily data from daily. For example a value of 3 would return data every 3 hours. Only divisors of 24 work (i.e. 1,2,3,4, etc.).
temp.units	Option to specify the units in which the temperature is entered. Default is Farenheit.
rh.units	Option to specify the units in which the relative humidity is entered. Default is percent.

ws.units	Option to specify the units in which the wind speed is entered. Default is miles per hour.
pp.units	Option to specify the units in which the precipitation is entered. Default is inches.
list()	additional arguments to be passed to lightME

---

weachDT *weachDT*

Description

Simple, Fast Daily to Hourly Climate Downscaling

Usage

```
weachDT(X, lati)
```

Arguments

X	data table with climate variables
lati	latitude (for calculating solar radiation)

Details

Based on weach family of functions but 5x faster than weachNEW, and requiring metric units (temperature in celsius, windspeed in kph, precip in mm, relative humidity as fraction)

Value

weather file for input to BioGro and related crop growth functions

Author(s)

David LeBauer

weachNEW *algorithm that decides weather weach365 or weach366 will be used based on the output of CheckLeapYear*

Description

algorithm that decides weather weach365 or weach366 will be used based on the output of CheckLeapYear

Usage

```
weachNEW(X, lati, ts = 1, temp.units = c("Farenheit", "Celsius"),
  rh.units = c("percent", "fraction"), ws.units = c("mph", "mps"),
  pp.units = c("in", "mm"), ...)
```

Arguments

X	a matrix (or data frame) containing weather information. The input format is strict but it is meant to be used with the data usually obtained from weather stations in Illinois. The data frame should have 11 columns (see details).
lati	latitude at the specific location
ts	timestep for the simulation of sub-daily data from daily. For example a value of 3 would return data every 3 hours. Only divisors of 24 work (i.e. 1,2,3,4, etc.).
temp.units	Option to specify the units in which the temperature is entered. Default is Farenheit.
rh.units	Option to specify the units in which the relative humidity is entered. Default is percent.
ws.units	Option to specify the units in which the wind speed is entered. Default is miles per hour.
pp.units	Option to specify the units in which the precipitation is entered. Default is inches.
list()	additional arguments to be passed to lightME

weach_imn *Weather change Iowa Mesonet*

Description

Manipulates weather data in the format obtained from Iowa Mesonet (see link below) and returns the format and units needed for most functions in this package. This function should be used for one year at a time. It takes and returns hourly weather information only.

Usage

```
weach_imn(data, lati, ts = 1, temp.units = c("Fahrenheit", "Celsius"),
  rh.units = c("percent", "fraction"), ws.units = c("mph", "mps"),
  pp.units = c("in", "mm"), ...)
```

Arguments

`data`	data as obtained from the Iowa Mesonet (see details)
`lati`	Latitude, not used at the moment
`ts`	Time step, not used at the moment
`temp.units`	Temperature units
`rh.units`	Relative humidity units
`ws.units`	wind speed units
`pp.units`	precipitation units
`...`	

Details

This function should be used to transform data from the Iowa Mesonet at hourly intervals from here:

http://mesonet.agron.iastate.edu/agclimate/hist/hourlyRequest.php

When selecting to download variables: Air Temperature (Fahrenheit) Solar Radiation (kilocalories per meter squared) Precipitation (inches) Relative humidity (percent) Wind Speed (mph)

You can read the data directly as it is downloaded making sure you skip the first 6 lines (This includes the title row).

The imported data frame should have 9 columns with:

1. site ID
2. site name
3. date in format "year-month-day", e.g. '2010-3-25'
4. hour in format "hour:minute", e.g. '15:00'
5. temperature (Fahrenheit)
6. solar radiation (kilocalories per meter squared)
7. precipitation (inches)
8. relative humidity (%).
9. wind speed (mph)

above~~

Value

It will return a data frame in the same format as the `weach` function.

Author(s)

Fernando E. Miguez

References

Iowa Mesonet http://mesonet.agron.iastate.edu/index.phtml

See Also

`weach` `help`, `~~~`

Examples

```
## Read an example data set from my website
url <- "http://www.agron.iastate.edu/miguezlab/teaching/CropSoilModel/ames_2010-iowamesonet.
ames.wea <- read.table(url, skip = 6)
ames.wea2 <- weach_imn(ames.wea)
```

`weather05` *Weather data*

Description

Weather data as produced by the `weach` function. These are for 2004 and 2005.

Format

data frame of dimensions 8760 by 7.

Source

simulated (based on Champaign, Illinois conditions).

`weather06` *Weather data*

Format

A data frame with 8760 observations on the following 8 variables.

list('year') a numeric vector
list('doy') a numeric vector
list('hour') a numeric vector
list('SolarR') a numeric vector
list('Temp') a numeric vector
list('RH') a numeric vector
list('WS') a numeric vector
list('precip') a numeric vector

Details

above ~~

Source

were obtained ~~

Examples

```
data(weather06)
## maybe str(weather06) ; plot(weather06) ...
```

---

willowCent *willowmass crops growth simulation*

Description

Simulates dry biomass growth during an entire growing season. It represents an integration of the photosynthesis function `c3photo`, canopy evapo/transpiration `CanA`, the multilayer canopy model `sunML` and a dry biomass partitioning calendar and senescence. It also considers, carbon and nitrogen cycles and water and nitrogen limitations.

Usage

```
willowCent(WetDat, day1 = 120, dayn = 300, timestep = 1, iRhizome = 1,
  lat = 40, iPlant = 1, irtl = 1e-04, canopyControl = list(),
  seneControl = list(), photoControl = list(),
  willowphenoControl = list(), soilControl = list(),
  nitroControl = list(), iPlantControl = list(), centuryControl = list())
```

Arguments

`WetDat`	weather data as produced by the `weach` function.
`day1`	first day of the growing season, (1–365).
`dayn`	last day of the growing season, (1–365, but larger than `day1`). See details.
`timestep`	Simulation timestep, the default of 1 requires houlry weather data. A value of 3 would require weather data every 3 hours. This number should be a divisor of 24.
`lat`	latitude, default 40.
`iRhizome`	initial dry biomass of the Rhizome (Mg ha^{-1}).
`irtl`	Initial rhizome proportion that becomes leaf. This should not typically be changed, but it can be used to indirectly control the effect of planting density.

`canopyControl`

List that controls aspects of the canopy simulation. It should be supplied through the `canopyParms` function.

`Sp` (specific leaf area) here the units are ha Mg^{-1}. If you have data in m^2 of leaf per kg of dry matter (e.g. 15) then divide by 10 before inputting this coefficient.

`nlayers` (number of layers of the canopy) Maximum 50. To increase the number of layers (more than 50) the `C` source code needs to be changed slightly.

`kd` (extinction coefficient for diffuse light) between 0 and 1.

`mResp` (maintenance respiration) a vector of length 2 with the first component for leaf and stem and the second component for rhizome and root.

`seneControl` List that controls aspects of senescence simulation. It should be supplied through the `seneParms` function.

`senLeaf` Thermal time at which leaf senescence will start.

`senStem` Thermal time at which stem senescence will start.

`senRoot` Thermal time at which root senescence will start.

`senRhizome` Thermal time at which rhizome senescence will start.

`photoControl` List that controls aspects of photosynthesis simulation. It should be supplied through the `photoParms` function.

`vmax` Vmax passed to the `c3photo` function.

`alpha` alpha parameter passed to the `c3photo` function.

`theta` theta parameter passed to the `c3photo` function.

`beta` beta parameter passed to the `c3photo` function.

`Rd` Rd parameter passed to the `c3photo` function.

`Catm` Catm parameter passed to the `c3photo` function.

`b0` b0 parameter passed to the `c3photo` function.

`b1` b1 parameter passed to the `c3photo` function.

`phenoControl` List that controls aspects of the crop phenology. It should be supplied through the `phenoParms` function.

`tp1-tp6` thermal times which determine the time elapsed between phenological stages. Between 0 and tp1 is the juvenile stage. etc.

`kLeaf1-6` proportion of the carbon that is allocated to leaf for phenological stages 1 through 6.

`kStem1-6` proportion of the carbon that is allocated to stem for phenological stages 1 through 6.

`kRoot1-6` proportion of the carbon that is allocated to root for phenological stages 1 through 6.

`kRhizome1-6` proportion of the carbon that is allocated to rhizome for phenological stages 1 through 6.

`kGrain1-6` proportion of the carbon that is allocated to grain for phenological stages 1 through 6. At the moment only the last stage (i.e. 6 or post-flowering) is allowed to be larger than zero. An error will be returned if kGrain1-5 are different from zero.

`soilControl` List that controls aspects of the soil environment. It should be supplied through the `soilParms` function.

FieldC Field capacity. This can be used to override the defaults possible from the soil types (see showSoilType).

WiltP Wilting point. This can be used to override the defaults possible from the soil types (see showSoilType).

phi1 Parameter which controls the spread of the logistic function. See wtrstr for more details.

phi2 Parameter which controls the reduction of the leaf area growth due to water stress. See wtrstr for more details.

soilDepth Maximum depth of the soil that the roots have access to (i.e. rooting depth).

iWatCont Initial water content of the soil the first day of the growing season. It can be a single value or a vector for the number of layers specified.

soilType Soil type, default is 6 (a more typical soil would be 3). To see details use the function showSoilType.

soilLayer Integer between 1 and 50. The default is 5. If only one soil layer is used the behavior can be quite different.

soilDepths Intervals for the soil layers.

wsFun one of 'logistic','linear','exp' or 'none'. Controls the method for the relationship between soil water content and water stress factor.

scsf stomatal conductance sensitivity factor (default = 1). This is an empirical coefficient that needs to be adjusted for different species.

rfl Root factor lambda. A Poisson distribution is used to simulate the distribution of roots in the soil profile and this parameter can be used to change the lambda parameter of the Poisson.

rsec Radiation soil evaporation coefficient. Empirical coefficient used in the incidence of direct radiation on soil evaporation.

rsdf Root soil depth factor. Empirical coefficient used in calculating the depth of roots as a function of root biomass.

nitroControl List that controls aspects of the nitrogen environment. It should be supplied through the nitrolParms function.

iLeafN initial value of leaf nitrogen (g m-2).

kLN coefficient of decrease in leaf nitrogen during the growing season. The equation is LN = iLeafN * (Stem + Leaf)^-kLN .

Vmax.b1 slope which determines the effect of leaf nitrogen on Vmax.

alpha.b1 slope which controls the effect of leaf nitrogen on alpha.

centuryControl

List that controls aspects of the Century model for carbon and nitrogen dynamics in the soil. It should be supplied through the centuryParms function.

SC1-9 Soil carbon pools in the soil. SC1: Structural surface litter. SC2: Metabolic surface litter. SC3: Structural root litter. SC4: Metabolic root litter. SC5: Surface microbe. SC6: Soil microbe. SC7: Slow carbon. SC8: Passive carbon. SC9: Leached carbon.

LeafL.Ln Leaf litter lignin content.

StemL.Ln Stem litter lignin content.

RootL.Ln Root litter lignin content.

`RhizomeL.Ln` Rhizome litter lignin content.
`LeafL.N` Leaf litter nitrogen content.
`StemL.N` Stem litter nitrogen content.
`RootL.N` Root litter nitrogen content.
`RhizomeL.N` Rhizome litter nitrogen content.
`Nfert` Nitrogen from a fertilizer source.
`iMinN` Initial value for the mineral nitrogen pool.
`Litter` Initial values of litter (leaf, stem, root, rhizome).
`timestep` currently either week (default) or day.

Value

a `list` structure with components

Examples

```
## Not run:
data(weather05)

res0 <- willowGro(weather05)

plot(res0)

## Looking at the soil model

res1 <- willowGro(weather05, soilControl = soilParms(soilLayers = 6))
plot(res1, plot.kind='SW') ## Without hydraulic distribution
res2 <- willowGro(weather05, soilControl = soilParms(soilLayers = 6, hydrDist=TRUE))
plot(res2, plot.kind='SW') ## With hydraulic distribution


## Example of user defined soil parameters.
## The effect of phi2 on yield and soil water content

ll.0 <- soilParms(FieldC=0.37,WiltP=0.2,phi2=1)
ll.1 <- soilParms(FieldC=0.37,WiltP=0.2,phi2=2)
ll.2 <- soilParms(FieldC=0.37,WiltP=0.2,phi2=3)
ll.3 <- soilParms(FieldC=0.37,WiltP=0.2,phi2=4)

ans.0 <- willowGro(weather05,soilControl=ll.0)
ans.1 <- willowGro(weather05,soilControl=ll.1)
ans.2 <- willowGro(weather05,soilControl=ll.2)
ans.3 <-willowGro(weather05,soilControl=ll.3)

xyplot(ans.0$SoilWatCont +
       ans.1$SoilWatCont +
       ans.2$SoilWatCont +
       ans.3$SoilWatCont ~ ans.0$DayofYear,
       type='l',
       ylab='Soil water Content (fraction)',
       xlab='DOY')
```

```
## Compare LAI

xyplot(ans.0$LAI +
       ans.1$LAI +
       ans.2$LAI +
       ans.3$LAI ~ ans.0$DayofYear,
       type='l',
       ylab='Leaf Area Index',
       xlab='DOY')



## End(Not run)
```

`willowGro` *willowmass crops growth simulation*

Description

Simulates dry biomass growth during an entire growing season. It represents an integration of the photosynthesis function `c3photo`, canopy evapo/transpiration `CanA`, the multilayer canopy model `sunML` and a dry biomass partitioning calendar and senescence. It also considers, carbon and nitrogen cycles and water and nitrogen limitations.

Usage

```
willowGro(WetDat, day1 = 120, dayn = 300, timestep = 1, iRhizome = 1,
  lat = 40, iPlant = 1, irtl = 1e-04, canopyControl = list(),
  seneControl = list(), photoControl = list(),
  willowphenoControl = list(), soilControl = list(),
  nitroControl = list(), iPlantControl = list(), centuryControl = list())
```

Arguments

`WetDat`	weather data as produced by the `weach` function.
`day1`	first day of the growing season, (1–365).
`dayn`	last day of the growing season, (1–365, but larger than `day1`). See details.
`timestep`	Simulation timestep, the default of 1 requires houlry weather data. A value of 3 would require weather data every 3 hours. This number should be a divisor of 24.
`lat`	latitude, default 40.
`iRhizome`	initial dry biomass of the Rhizome (Mg ha^{-1}).
`irtl`	Initial rhizome proportion that becomes leaf. This should not typically be changed, but it can be used to indirectly control the effect of planting density.

`canopyControl`

List that controls aspects of the canopy simulation. It should be supplied through the `canopyParms` function.

`Sp` (specific leaf area) here the units are ha Mg^{-1}. If you have data in m^2 of leaf per kg of dry matter (e.g. 15) then divide by 10 before inputting this coefficient.

`nlayers` (number of layers of the canopy) Maximum 50. To increase the number of layers (more than 50) the `C` source code needs to be changed slightly.

`kd` (extinction coefficient for diffuse light) between 0 and 1.

`mResp` (maintenance respiration) a vector of length 2 with the first component for leaf and stem and the second component for rhizome and root.

`seneControl` List that controls aspects of senescence simulation. It should be supplied through the `seneParms` function.

`senLeaf` Thermal time at which leaf senescence will start.

`senStem` Thermal time at which stem senescence will start.

`senRoot` Thermal time at which root senescence will start.

`senRhizome` Thermal time at which rhizome senescence will start.

`photoControl` List that controls aspects of photosynthesis simulation. It should be supplied through the `photoParms` function.

`vmax` Vmax passed to the `c3photo` function.

`alpha` alpha parameter passed to the `c3photo` function.

`theta` theta parameter passed to the `c3photo` function.

`beta` beta parameter passed to the `c3photo` function.

`Rd` Rd parameter passed to the `c3photo` function.

`Catm` Catm parameter passed to the `c3photo` function.

`b0` b0 parameter passed to the `c3photo` function.

`b1` b1 parameter passed to the `c3photo` function.

`phenoControl` List that controls aspects of the crop phenology. It should be supplied through the `phenoParms` function.

`tp1-tp6` thermal times which determine the time elapsed between phenological stages. Between 0 and tp1 is the juvenile stage. etc.

`kLeaf1-6` proportion of the carbon that is allocated to leaf for phenological stages 1 through 6.

`kStem1-6` proportion of the carbon that is allocated to stem for phenological stages 1 through 6.

`kRoot1-6` proportion of the carbon that is allocated to root for phenological stages 1 through 6.

`kRhizome1-6` proportion of the carbon that is allocated to rhizome for phenological stages 1 through 6.

`kGrain1-6` proportion of the carbon that is allocated to grain for phenological stages 1 through 6. At the moment only the last stage (i.e. 6 or post-flowering) is allowed to be larger than zero. An error will be returned if kGrain1-5 are different from zero.

`soilControl` List that controls aspects of the soil environment. It should be supplied through the `soilParms` function.

FieldC Field capacity. This can be used to override the defaults possible from the soil types (see showSoilType).

WiltP Wilting point. This can be used to override the defaults possible from the soil types (see showSoilType).

phi1 Parameter which controls the spread of the logistic function. See wtrstr for more details.

phi2 Parameter which controls the reduction of the leaf area growth due to water stress. See wtrstr for more details.

soilDepth Maximum depth of the soil that the roots have access to (i.e. rooting depth).

iWatCont Initial water content of the soil the first day of the growing season. It can be a single value or a vector for the number of layers specified.

soilType Soil type, default is 6 (a more typical soil would be 3). To see details use the function showSoilType.

soilLayer Integer between 1 and 50. The default is 5. If only one soil layer is used the behavior can be quite different.

soilDepths Intervals for the soil layers.

wsFun one of 'logistic','linear','exp' or 'none'. Controls the method for the relationship between soil water content and water stress factor.

scsf stomatal conductance sensitivity factor (default = 1). This is an empirical coefficient that needs to be adjusted for different species.

rfl Root factor lambda. A Poisson distribution is used to simulate the distribution of roots in the soil profile and this parameter can be used to change the lambda parameter of the Poisson.

rsec Radiation soil evaporation coefficient. Empirical coefficient used in the incidence of direct radiation on soil evaporation.

rsdf Root soil depth factor. Empirical coefficient used in calculating the depth of roots as a function of root biomass.

nitroControl List that controls aspects of the nitrogen environment. It should be supplied through the nitrolParms function.

iLeafN initial value of leaf nitrogen (g m-2).

kLN coefficient of decrease in leaf nitrogen during the growing season. The equation is LN = iLeafN * (Stem + Leaf)^-kLN .

Vmax.b1 slope which determines the effect of leaf nitrogen on Vmax.

alpha.b1 slope which controls the effect of leaf nitrogen on alpha.

centuryControl

List that controls aspects of the Century model for carbon and nitrogen dynamics in the soil. It should be supplied through the centuryParms function.

SC1-9 Soil carbon pools in the soil. SC1: Structural surface litter. SC2: Metabolic surface litter. SC3: Structural root litter. SC4: Metabolic root litter. SC5: Surface microbe. SC6: Soil microbe. SC7: Slow carbon. SC8: Passive carbon. SC9: Leached carbon.

LeafL.Ln Leaf litter lignin content.

StemL.Ln Stem litter lignin content.

RootL.Ln Root litter lignin content.

RhizomeL.Ln Rhizome litter lignin content.
LeafL.N Leaf litter nitrogen content.
StemL.N Stem litter nitrogen content.
RootL.N Root litter nitrogen content.
RhizomeL.N Rhizome litter nitrogen content.
Nfert Nitrogen from a fertilizer source.
iMinN Initial value for the mineral nitrogen pool.
Litter Initial values of litter (leaf, stem, root, rhizome).
timestep currently either week (default) or day.

Value

a list structure with components

Examples

```
## Not run:
data(weather05)

res0 <- willowGro(weather05)

plot(res0)

## Looking at the soil model

res1 <- willowGro(weather05, soilControl = soilParms(soilLayers = 6))
plot(res1, plot.kind='SW') ## Without hydraulic distribution
res2 <- willowGro(weather05, soilControl = soilParms(soilLayers = 6, hydrDist=TRUE))
plot(res2, plot.kind='SW') ## With hydraulic distribution


## Example of user defined soil parameters.
## The effect of phi2 on yield and soil water content

ll.0 <- soilParms(FieldC=0.37,WiltP=0.2,phi2=1)
ll.1 <- soilParms(FieldC=0.37,WiltP=0.2,phi2=2)
ll.2 <- soilParms(FieldC=0.37,WiltP=0.2,phi2=3)
ll.3 <- soilParms(FieldC=0.37,WiltP=0.2,phi2=4)

ans.0 <- willowGro(weather05,soilControl=ll.0)
ans.1 <- willowGro(weather05,soilControl=ll.1)
ans.2 <- willowGro(weather05,soilControl=ll.2)
ans.3 <-willowGro(weather05,soilControl=ll.3)

xyplot(ans.0$SoilWatCont +
       ans.1$SoilWatCont +
       ans.2$SoilWatCont +
       ans.3$SoilWatCont ~ ans.0$DayofYear,
       type='l',
       ylab='Soil water Content (fraction)',
       xlab='DOY')
```

```
## Compare LAI

xyplot(ans.0$LAI +
       ans.1$LAI +
       ans.2$LAI +
       ans.3$LAI ~ ans.0$DayofYear,
       type='l',
       ylab='Leaf Area Index',
       xlab='DOY')



## End(Not run)
```

`wsRcoef` *R coefficient for water stress*

Description

determines whether the argument wsFun is linear, logistic, exponential, or something else and calculates a value for wsPhoto based on that.

Usage

```
wsRcoef(aw, fieldc, wiltp, phi1, phi2, wsFun = c("linear", "logistic", "exp",
  "none"))
```

Arguments

`aw`	available water
`fieldc`	Field capacity of the soil (fraction).
`wiltp`	Wilting point of the soil (fraction).
`phi1`	coefficient which controls the spread of the logistic function.
`phi2`	coefficient which controls the effect on leaf area expansion.
`wsFun`	option to control which method is used for the water stress function.

`wtrstr` *Simple function to illustrate soil water content effect on plant water stress.*

Description

This is a very simple function which implements the 'bucket' model for soil water content and it calculates a coefficient of plant water stress.

Usage

```
wtrstr(precipt, evapo, cws, soildepth, fieldc, wiltp, phi1 = 0.01,
  phi2 = 10, wsFun = c("linear", "logistic", "exp", "none"))
```

Arguments

`precipt`	Precipitation (mm).
`evapo`	Evaporation (Mg H2O ha-1 hr-1).
`cws`	current water content (fraction).
`soildepth`	Soil depth, typically 1m.
`fieldc`	Field capacity of the soil (fraction).
`wiltp`	Wilting point of the soil (fraction).
`phi1`	coefficient which controls the spread of the logistic function.
`phi2`	coefficient which controls the effect on leaf area expansion.
`wsFun`	option to control which method is used for the water stress function.

Details

This is a very simple function and the details can be seen in the code.

Value

A list with components:

See Also

`wsRcoef`

Examples

```
## Looking at the three possible models for the effect of soil moisture on water
## stress

aws <- seq(0,0.4,0.001)
wats.L <- numeric(length(aws)) # linear
wats.Log <- numeric(length(aws)) # logistic
```

```
wats.exp <- numeric(length(aws)) # exp
wats.none <- numeric(length(aws)) # none
for(i in 1:length(aws)){
wats.L[i] <- wtrstr(1,1,aws[i],0.5,0.37,0.2,2e-2,4)$wsPhoto
wats.Log[i] <- wtrstr(1,1,aws[i],0.5,0.37,0.2,2e-2,4,wsFun='logistic')$wsPhoto
wats.exp[i] <- wtrstr(1,1,aws[i],0.5,0.37,0.2,2e-2,4, wsFun='exp')$wsPhoto
wats.none[i] <- wtrstr(1,1,aws[i],0.5,0.37,0.2,2e-2,4, wsFun='none')$wsPhoto
}

xyplot(wats.L + wats.Log + wats.exp  + wats.none~ aws,
       col=c('blue','green','purple','red'),
       type = 'l',
       xlab='Soil Water',
       ylab='Stress Coefficient',
       key = list(text=list(c('linear','logistic','exp', 'none')),
       col=c('blue','green','purple','red'), lines = TRUE) )

## This function is sensitive to the soil depth parameter

SDepth <- seq(0.05,2,0.05)

wats <- numeric(length(SDepth))

for(i in 1:length(SDepth)){
wats[i] <- wtrstr(1,1,0.3,SDepth[i],0.37,0.2,2e-2,3)$wsPhoto
}

xyplot(wats ~ SDepth, ylab='Water Stress Coef',
       xlab='Soil depth')

## Difference between the effect on assimilation and leaf expansion rate

aws <- seq(0,0.4,0.001)
wats.P <- numeric(length(aws))
wats.L <- numeric(length(aws))
for(i in 1:length(aws)){
wats.P[i] <- wtrstr(1,1,aws[i],0.5,0.37,0.2,2e-2,4)$wsPhoto
wats.L[i] <- wtrstr(1,1,aws[i],0.5,0.37,0.2,2e-2,4)$wsSpleaf
}

xyplot(wats.P + wats.L ~ aws,
       xlab='Soil Water',
       ylab='Stress Coefficient')


## An example for wsRcoef
## The scale parameter makes a big difference

aws <- seq(0.2,0.4,0.001)
wats.1 <- wsRcoef(aw=aws,fieldc=0.37,wiltp=0.2,phi1=1e-2,phi2=1, wsFun='logistic')$wsPhoto
wats.2 <- wsRcoef(aw=aws,fieldc=0.37,wiltp=0.2,phi1=2e-2,phi2=1, wsFun='logistic')$wsPhoto
wats.3 <- wsRcoef(aw=aws,fieldc=0.37,wiltp=0.2,phi1=3e-2,phi2=1, wsFun='logistic')$wsPhoto
```

```
xyplot(wats.1 + wats.2 + wats.3 ~ aws,type='l',
       col=c('blue','red','green'),
       ylab='Water Stress Coef',
       xlab='SoilWater Content',
       key=list(text=list(c('phi1 = 1e-2','phi1 = 2e-2','phi1 = 3e-2')),
         lines=TRUE,col=c('blue','red','green')))
```