Simulation and Parameter Estimation for Biomass Crops

Fernando E. Miguez
Department of Agronomy
Iowa State University

April 2, 2015

Abstract

Simulation and parameter estimation of photosynthesis and crop growth. The interest in developing this model is to be able to efficiently perform simulations of photosynthesis and crop growth. Since often this requires running a model multiple times, R provides a nice environment for optimization and plotting. The package also has a soil carbon and nitrogen model based on the Century model and a simple multilayered water soil model. As with many crop models the objective is to improve our understanding of productivity and carbon, water and nitrogen cycles in agro-ecosystems.

Contents

1 Introduction

The package BioCro started as a way to continue work on the ideas developed in the WIMOVAC model. WIMOVAC was developed by Stephen Long and Steve Humphries `http://www.life.illinois.edu/plantbio/wimovac/` and there have been several publications using this model. I have used the model for some initial efforts at modeling *Miscanthus* × *giganteus*.

This vignette was created using BioCro package version

```
sessionInfo()$otherPkgs$BioCro$Version

## [1] "0.267-1"
```

2 Carbon: Leaf-level Photosynthesis

There is a large range in the complexity of models that have been used to simulate photosynthesis. BioCro offers functions `c4photo` and `c3photo`. Both functions take as minimum input radiation (PAR $\mu mol\ m^{-2}\ s^{-1}$), temperature (Celsius) and relative humidity (0-1).

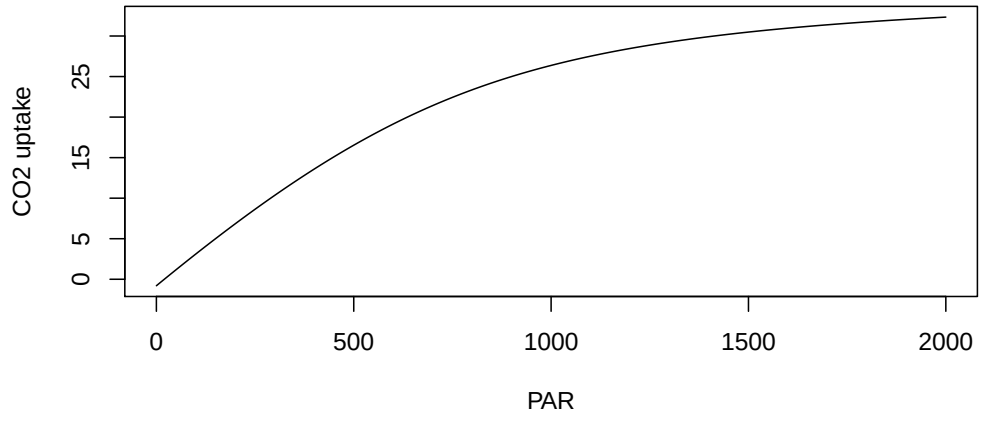
Since WIMOVAC originated as a photosynthesis model we can start with a simple example. For the C_4 photosynthesis model the best reference is Collatz et al. (1992).

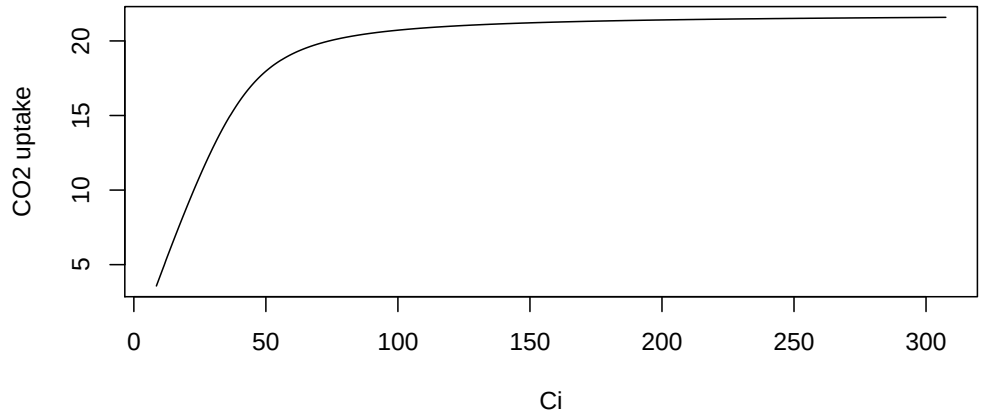
```
c4photo(1500,25,0.7)
```

```
## $Gs
## [1] 277.9524
##
## $Assim
## [1] 30.48496
##
## $Ci
## [1] 204.517
```

This example shows that with PAR 1500, temperature of 25 and relative humidity of 0.7 (70%) as inputs we get simulation of CO_2 assimilation (`Assim`), stomatal conductance (`Gs`) and the intercellular CO_2 (`Ci`). For units and other details see `?c4photo`. Another model available for simulating C4 photosynthesis is `eC4photo`, see docs for details. This model has not been used much. In `c4photo` the computation is carried out in compiled `C`, but there is a pure R function `c4photoR` which might be useful for understanding the calculations.

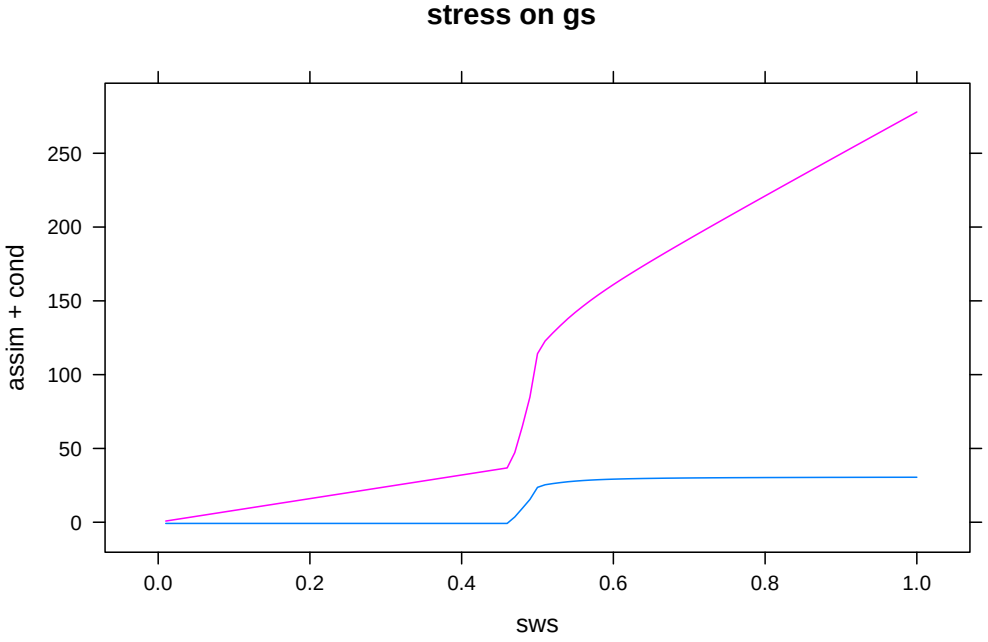
There is an equivalent function `c3photo` which is closely based on the c3 photosynthesis model described in WIMOVAC. Notice that the parameters and interpretation are different from the `c4photo` function.

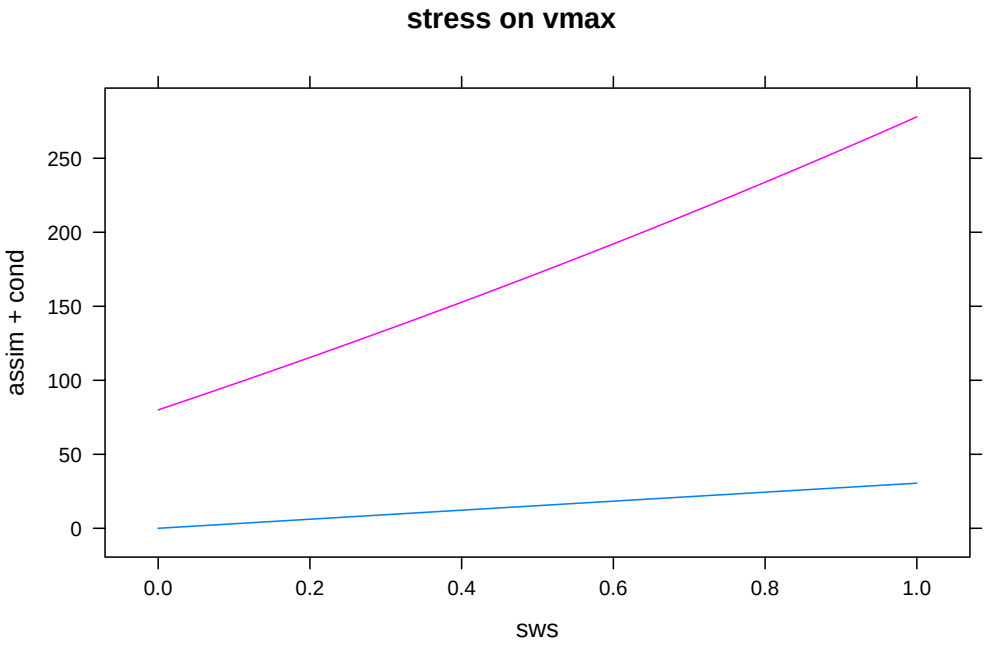




2.0.1 Effect of stress on photosynthesis

Besides some of the typical parameters for both functions there is the option of including stress. The argument is `stress`. The stress can be applied to stomatal conductance (default) or V_{max}.





2.1 Estimating photosynthesis parameters

The `c4photo` function has three main parameters that can be adjusted for different species or environments. These three parameters are V_{max} (light-saturated rate of photosynthesis which for C_4 plants is equivalent to the maximum rate of carboxylation, $\mu\ mol\ m^{-2}\ s^{-1}$), *alpha* (quantum efficiency, $mol\ mol^{-}1$) and dark respiration (net CO_2 exchange at zero light, $\mu\ mol\ m^{-2}\ s^{-1}$).

Options for adjusting the parameters for simulation of biomass crops:

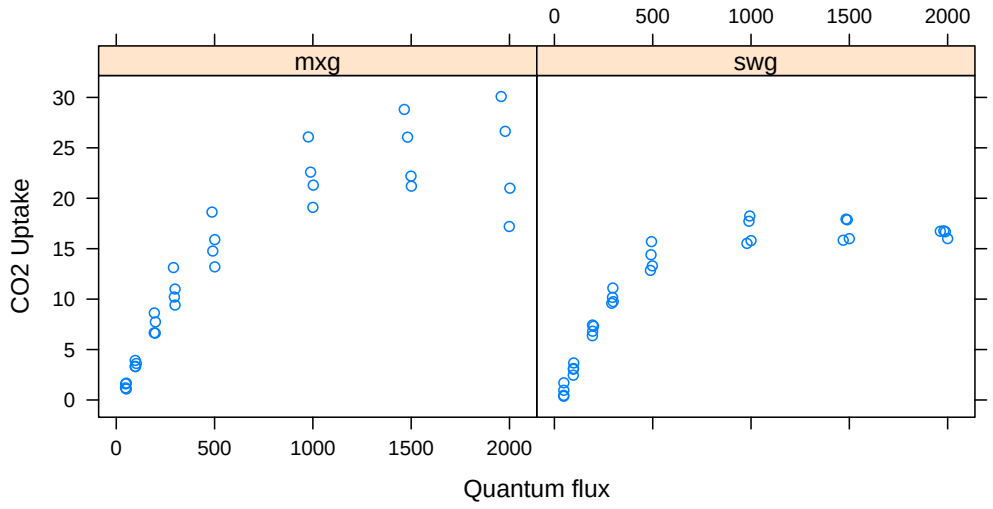
- Use values from the literature
- Optimize parameter values based on observed data

Using values from the literature can work well most of the time, but the source of the values should be carefully assessed. For example, an important consideration is whether the values were reported at the same reference temperature. In BioCro these values are assumed to be at 25 C. Another consideration is that different models might have different parameterization

of photosynthesis and what might work well in a different model might not work well in BioCro.

For optimization of these parameters based on observed data BioCro offers the ability to optimize these three parameters using different techniques. The first one is based on the `optim` function in R which will minimize the residual sum of squares (obs - sim). The first column has the curve ID, the second one the treatment which in this case is either Miscanthus (mxg) or switchgrass (swg). The third column is CO_2 assimilation, then photosynthetic active radiation in (PARin), temperature (Tleaf) and relative humidity (RH_S).

```
data(aq)
head(aq)

##   ID trt     A    PARi Tleaf RH_S
## 1  1 swg 16.68 1988.36 26.88 0.47
## 2  1 swg 17.87 1489.37 26.16 0.47
## 3  1 swg 18.24  993.60 25.39 0.47
## 4  1 swg 15.70  493.78 24.66 0.47
## 5  1 swg 11.10  297.23 24.34 0.47
## 6  1 swg  7.43  193.86 24.39 0.46

plotAQ(aq, type="p")
```



We could find out what the 'best' parameters are for the first curve.

```
curve1 <- subset(aq, ID == 1)
op <- Opc4photo(curve1[,3:6])
op

##
## Optimization of C4 photosynthesis
##
##   95 %   Conf Int
##           best    lower    upper
## Vmax    19.094   17.622   20.566
## alpha    0.052    0.042    0.062
##
##  Corr  Vmax and alpha: -0.3389846
##
##  Resid Sums Sq: 13.53705
##
## Convergence:YES
```

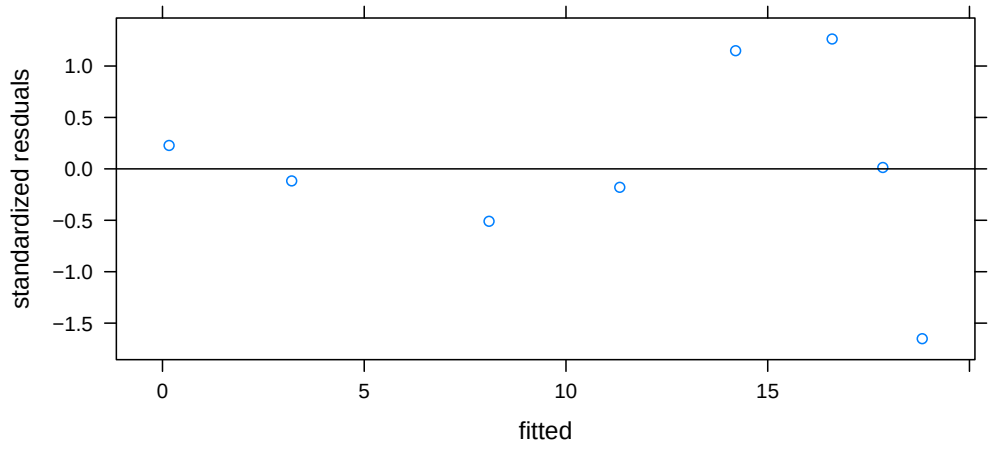
By default only V_{max} and *alpha* are optimized. *Rd* can also be optimized by setting the optimization level to 2.

```
op <- Opc4photo(curve1[,3:6], op.level=2)
op

##
## Optimization of C4 photosynthesis
##
##   95 %   Conf Int
##          best    lower    upper
## Vmax    21.47   18.527   24.416
## alpha    0.07    0.045    0.095
## Rd       3.18    0.485    5.878
##
##  Corr  Vmax and alpha: 0.7359342
##
##  Resid Sums Sq: 10.16169
```

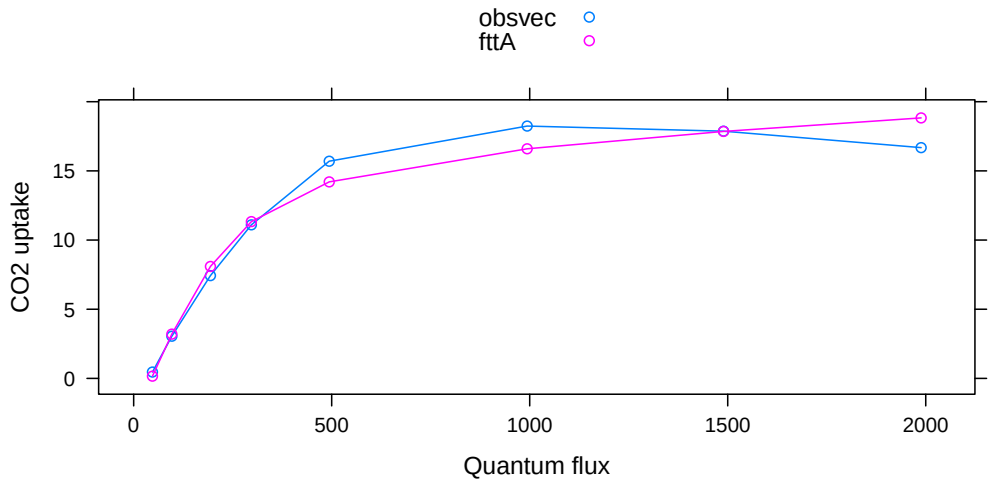
```
##
## Convergence:YES
```

There is also a plotting function for the `op` object which is of class "Opc4photo". This allows for examination of residuals and also comparing observed and simulated.

```
plot(op)
```




```
plot(op, plot.kind="OandF", type="o")
```

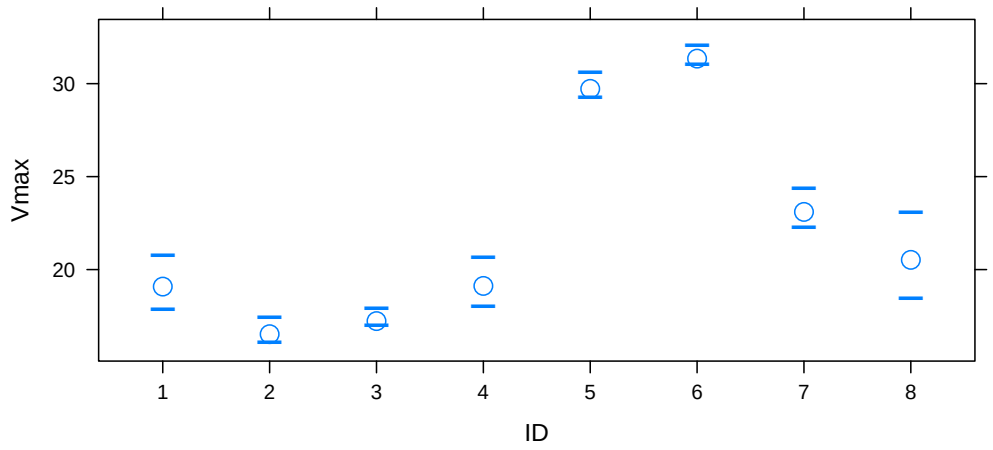


The previous example was for optimizing and analyzing a single curve. If we want to optimize several curves in parallel then the `mOpc4photo` function is available. For the following example we need to also supply a column with ambient CO_2 levels (in this case 390 ppm).

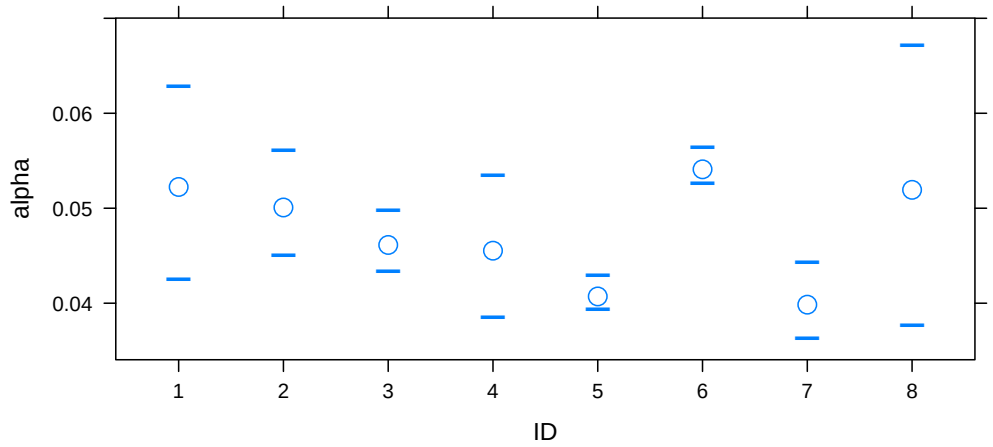
```
aq2 <- data.frame(aq[,-2], Catm=390)
mop <- mOpc4photo(aq2, verbose=TRUE)

## Run: 1 ... Converged YES
## Run: 2 ... Converged YES
## Run: 3 ... Converged YES
## Run: 4 ... Converged YES
## Run: 5 ... Converged YES
## Run: 6 ... Converged YES
## Run: 7 ... Converged YES
## Run: 8 ... Converged YES

mop

## Number of runs: 8
## Number converged: 8
##
##                mean           min           max
## vmax    22.0831619   16.52728571   31.34827910
## alpha    0.0475748    0.03984886    0.05410031
```

As with the previous example there are plotting methods. In this case the plotting functions are useful for visualizing the mean point estimate and the confidence intervals.

```
plot(mop)
```




```
plot(mop, parm="alpha")
```



As with the `Opc4photo` the optimization level can be changed to also optimize *Rd*.

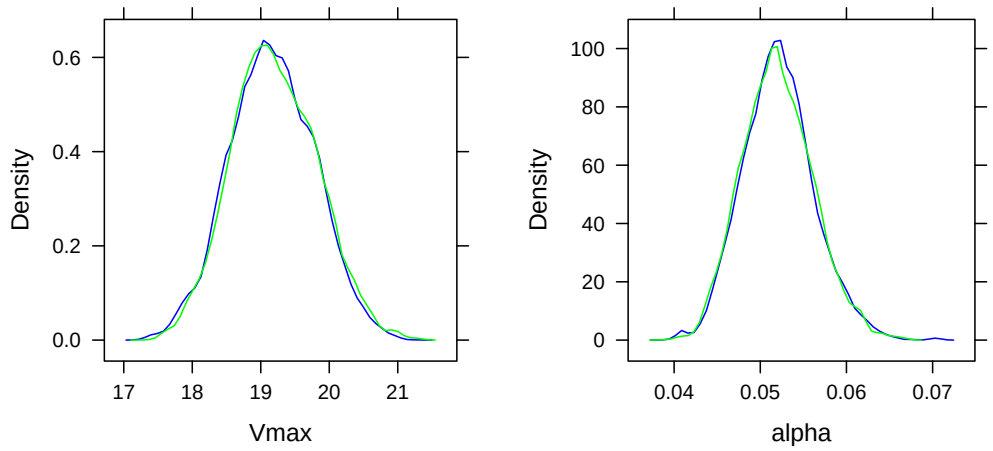
```
mop2 <- mOpc4photo(aq2, verbose=TRUE, op.level=2)

## Run: 1 ... Converged YES
## Run: 2 ... Converged YES
## Run: 3 ... Converged YES
```

```
## Run: 4 ... Converged YES
## Run: 5 ... Converged YES
## Run: 6 ... Converged YES
## Run: 7 ... Converged YES
## Run: 8 ... Converged YES

mop2

## Number of runs: 8
## Number converged: 8
##
##                mean           min           max
## vmax    22.92220989   16.61107519   31.79925899
## alpha    0.05359601    0.03623291    0.07002408
## Rd       1.63529875    0.06213858    3.18283189
```

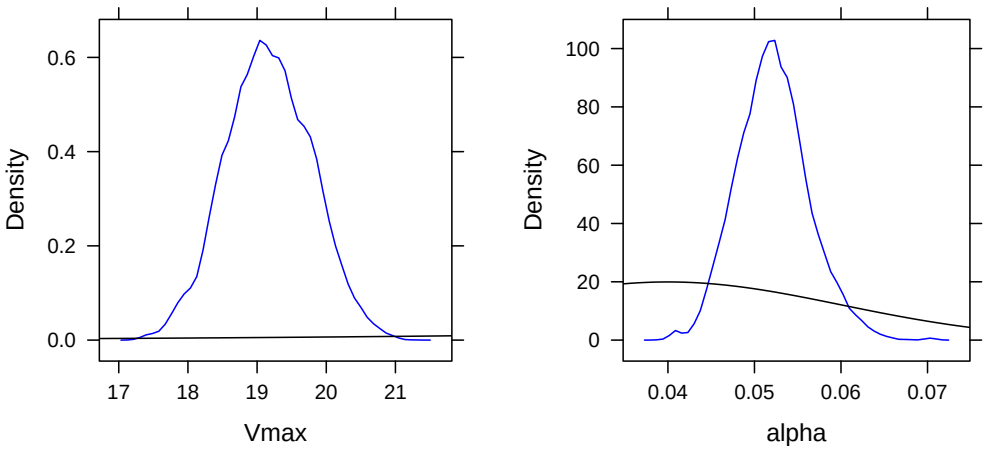
Yet another approach to optimizing photosynthetic parameters is to use a Bayesian approach where a prior distribution for the parameters can be specified. This approach might work better when there is limited data or when the previous approaches fail. The results are very similar to the example using `Opc4photo` for a single curve. In particular because in this case very diffuse priors were specified. The scale can be increased to reduce the acceptance rate. The function can also be run more than once to check the results.

```
op.mc1 <- MCMCc4photo(curve1[,3:6], scale=1.5)
op.mc2 <- MCMCc4photo(curve1[,3:6], scale=1.5)
op.mc1

##
##  Markov chain Monte Carlo for the Collatz C4 photosynthesis model
##  Summary of the chain
##  Moves: 9518 Prop: 0.4759
##
##  Summaries for vmax and alpha:
##           Min.  1st Qu.  Median     Mean  3rd Qu.     Max.
## vmax  17.07000 18.72000 19.1700 19.22000 19.62000 38.76000
## alpha  0.01778  0.04916  0.0519  0.05201  0.05472  0.07073
```

```
##
##  95 % Quantile Intervals for vmax and alpha:
##              0.025       0.975
## vmax  17.90783214 20.47381003
## alpha  0.04403637  0.06098266
##
##  correlation matrix:
##              Vmax      alpha
## Vmax   1.0000000 -0.4782492
## alpha -0.4782492  1.0000000
##
##  RSS range: 13.53716 595.9007

## plot(op.mc1, op.mc2) try this at home
plot(op.mc1, op.mc2, plot.kind="density", burnin=1e4)
```

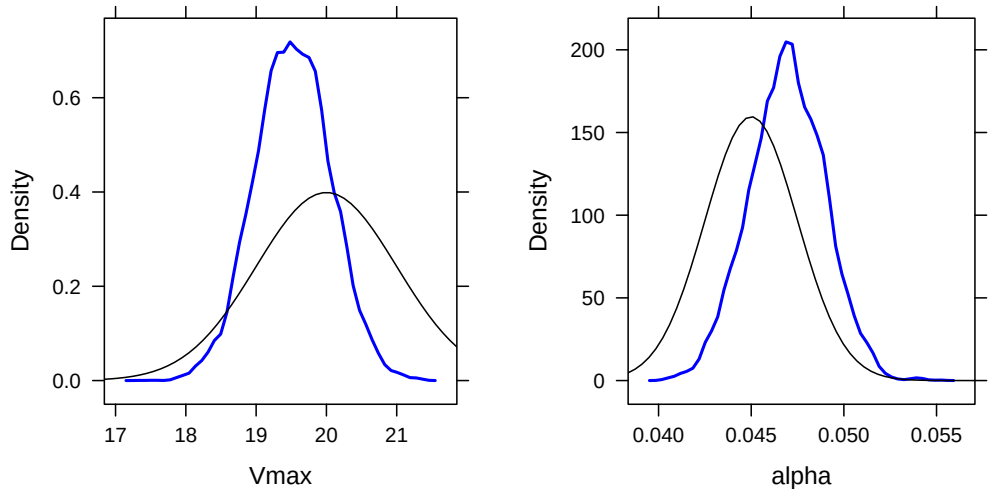



```
plot(op.mc1, plot.kind="density", prior=TRUE, burnin=1e4)
```



`prior=TRUE` plots the prior along side the results from the MCMC run. In this case the prior is very diffuse, but tighter priors would affect the results. If more stringent priors are set the results will tend to be closer to the prior distributions (the prior distributions are the black lines in the graph). In many cases a Bayesian approach is more reasonable as fitting individual curves in isolation can lead to values outside the range of reasonable results. The influence of the prior will lead the estimation closer to 'known' values for a species.

```
op.mc1 <- MCMCc4photo(curve1[,3:6], scale=1.5, prior=c(20, 1, 0.045, 0.0025))
plot(op.mc1, plot.kind = "density", prior = TRUE, burnin=1e3, lwd=2)
```



TODO: I haven't implemented the possibility of fitting Rd using the Bayesian approach. This is something I'd like to implement soon. Two other additions would be including the correlation between parameters and the estimation of the residual variance. These last features are low priority as the function as it is should work for most purposes.

To be complete there is even the option of using `nls`.

```
c4photo2 <- function(A,T,RH, vmax=39, alpha=0.04){
    res <- c4photo(A,T,RH, vmax=vmax, alpha=alpha)$Assim
    res
}
fit <- nls(A ~ c4photo2(PARi, Tleaf, RH_S, vmax, alpha),
           start=list(vmax=39, alpha=0.04),
           data = curve1)
```

TODO: run an example using `nlme`. Until I get this to work, the best option is to do a linear mixed model analysis using `lme`.

This concludes the section about estimating photosynthesis parameters in the context of simulating biomass crops. Although this is the first step, and it is important, there are many other aspects that influence the simulation of biomass, transpiration, etc.

The previous functions are relevant for leaf-level photosynthesis. Scaling up to the canopy level is not trivial since it requires developing a light macro

environment which simulates the partitioning between direct and diffuse radiation (see function `lightME`). The function `sunML` is used to predict the proportion of light for each layer of a multiple layered canopy.

TODO

- include an example using `c3photo`
- Discuss meaning and relationship among parameters

3 Carbon and Water: Canopy Photosynthesis and Transpiration

The function `CanA` integrates the previous functions to simulate canopy CO_2 assimilation for a complete canopy. This function also simulates transpiration based on Penman-Monteith, Penman and Priestly.

The `CanA` function is designed to run at an hourly timestep. The inputs should all be of length 1. As with other canopy models the canopy is discretized in layers and each layer has unique conditions in terms of light levels, leaf temperature, wind and relative humidity. See `?CanA` for details.

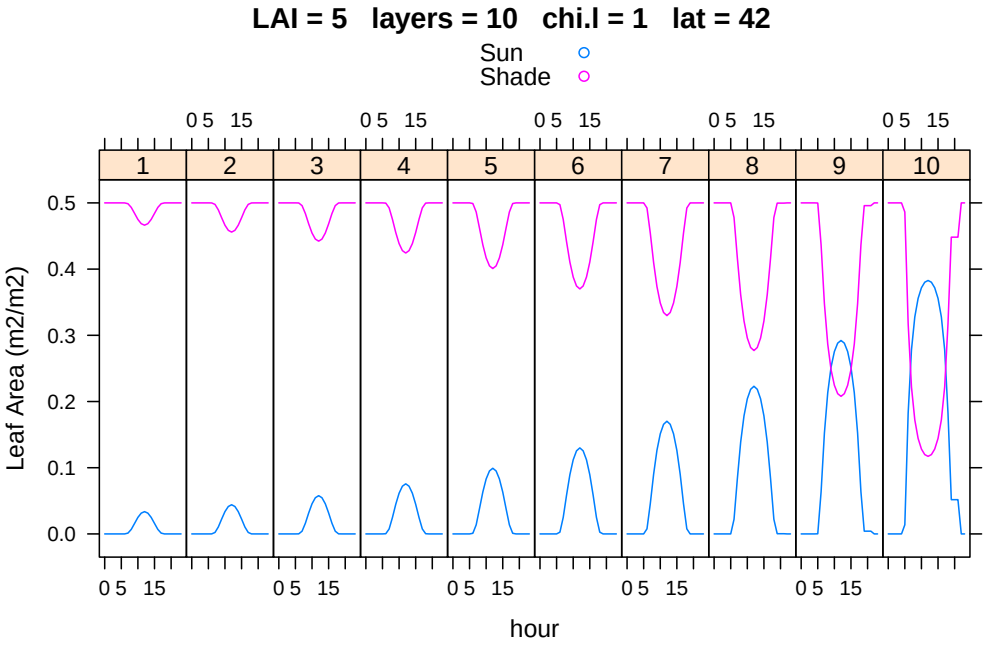
```
nlay <- 8
res <- CanA(lai=3, doy=200, hr=12, solar=1500, temp=25, rh=0.7,
            windspeed=2, nlayers=nlay)
```

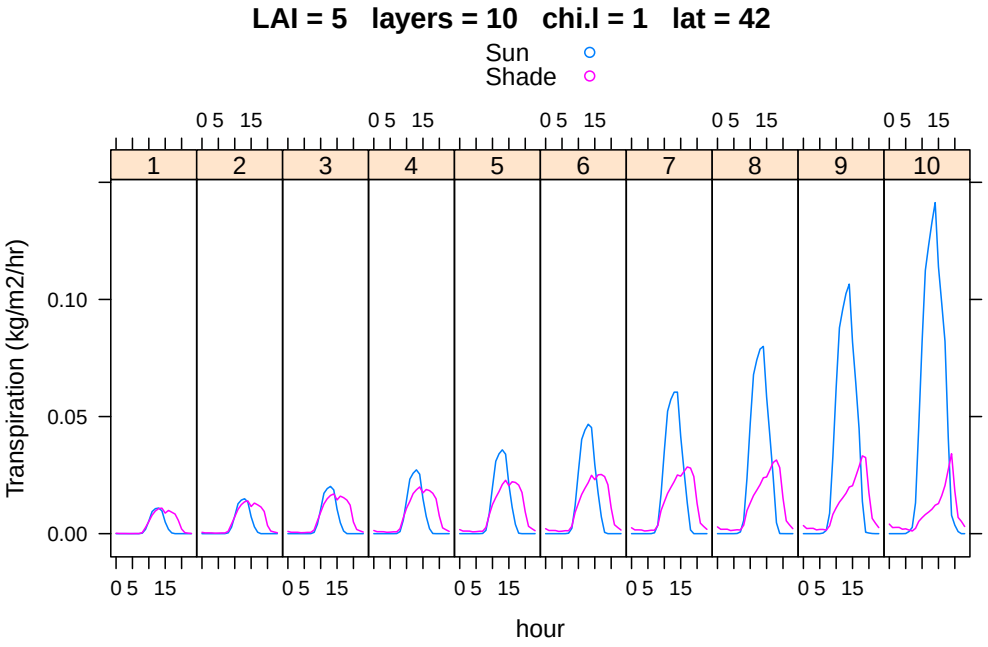
The distribution of leaves in the sun and in the shade is an important characteristic of a canopy and the architecture can be modified mainly by changing the `chi.l` parameter which represents the ratio of horizontal leaf projections to vertical leaf projections.

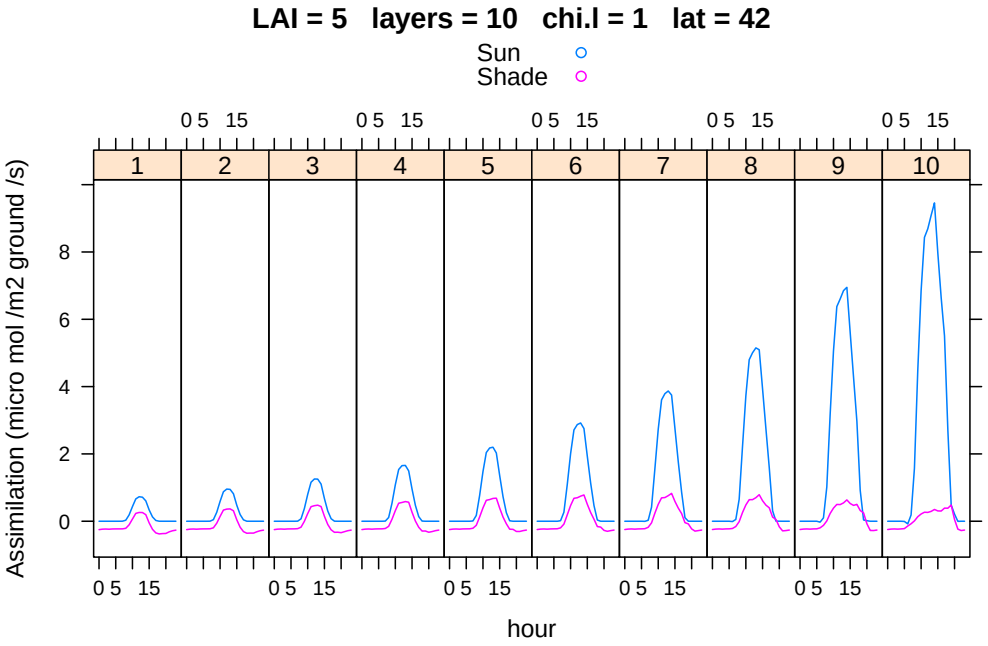
```
apply(res$LayMat[,3:4], 2, sum)

##   Leafsun Leafshade
##  1.354043  1.645957
```

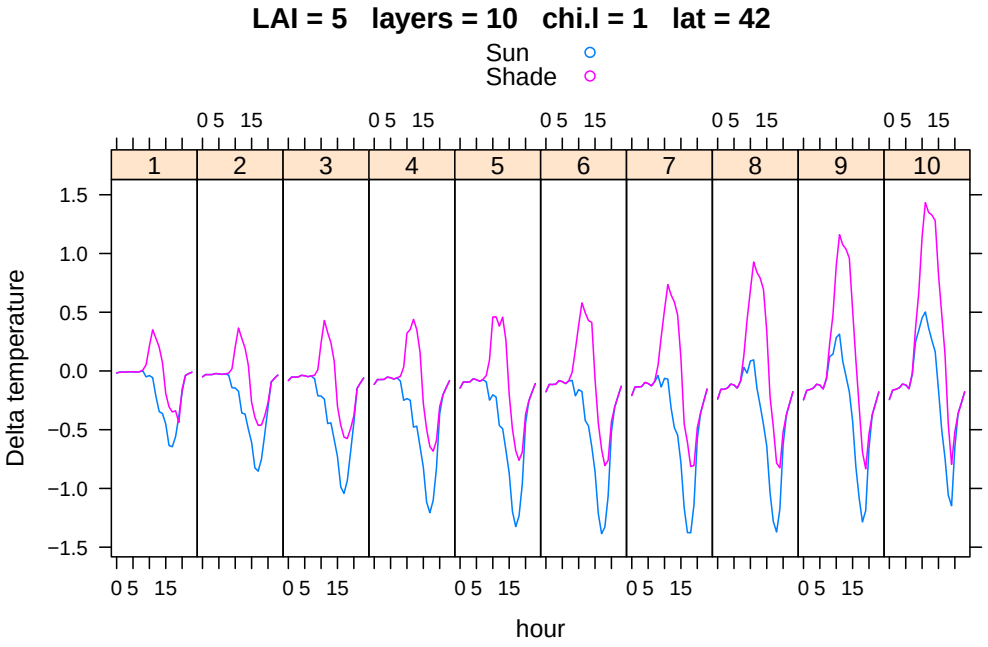
In this example, 1.35 m^2 of leaf are in the sun and 1.65 of m^2 are in the shade for a total of 3 LAI.

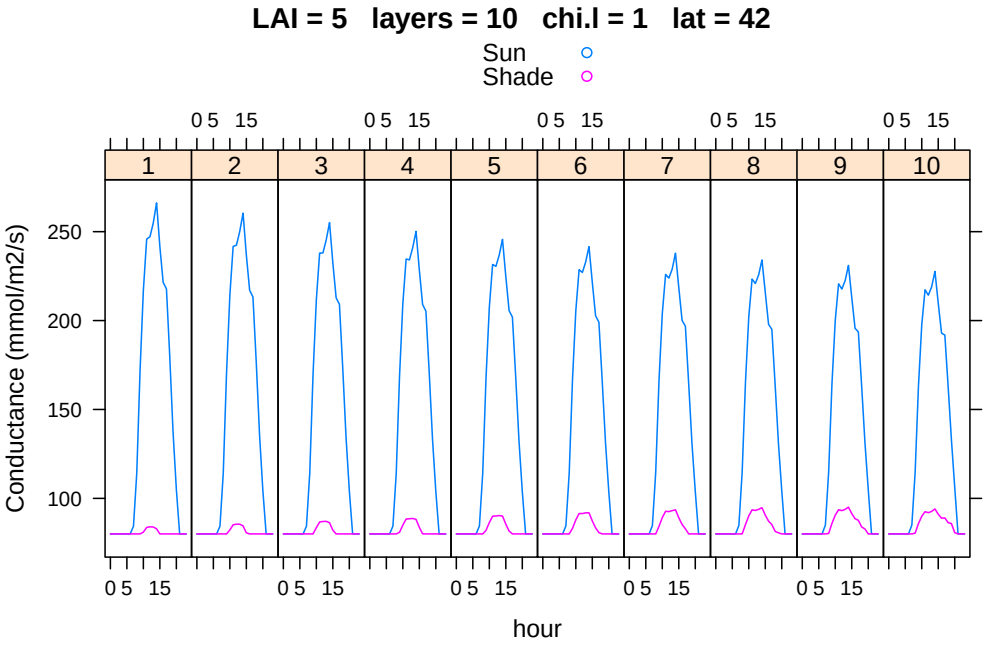
Next we can look in detail about the properties of the canopy by layer

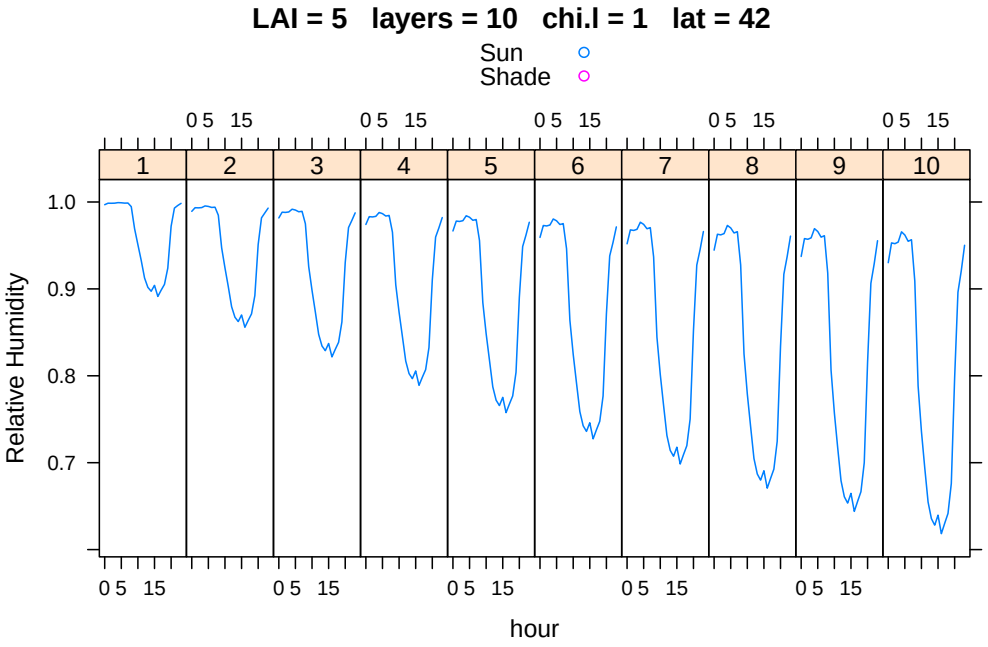

LAI = 5 layers = 10 chi.l = 1 lat = 42
Sun
Shade
1
2
3
4
5
6
7
8
9
10
0 5 15
0.0
0.1
0.2
0.3
0.4
0.5
Leaf Area (m2/m2)
hour

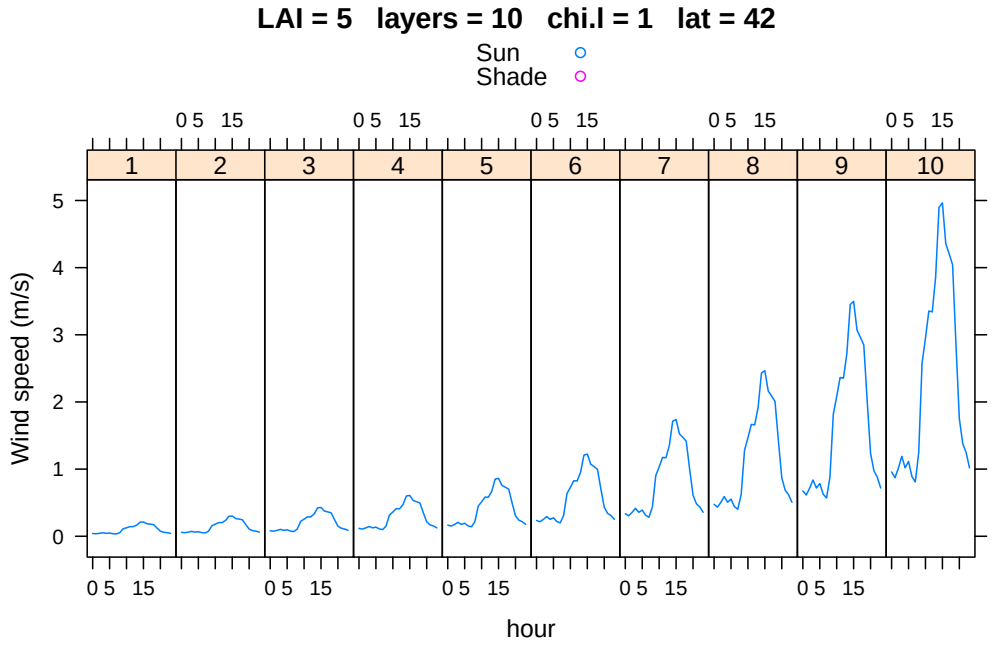

LAI = 5 layers = 10 chi.l = 1 lat = 42
Sun
Shade
1
2
3
4
5
6
7
8
9
10
0 5 15
0.00
0.05
0.10
Transpiration (kg/m2/hr)
hour

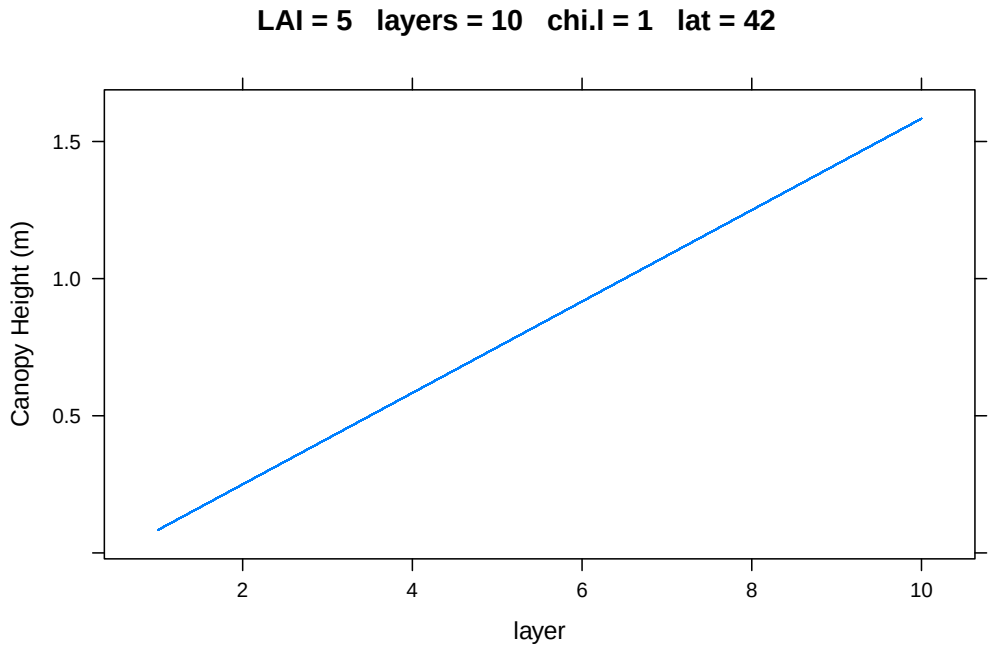

LAI = 5 layers = 10 chi.l = 1 lat = 42
Sun
Shade
1
2
3
4
5
6
7
8
9
10
Assimilation (micro mol /m2 ground /s)
hour


LAI = 5 layers = 10 chi.l = 1 lat = 42
Sun
Shade
1
2
3
4
5
6
7
8
9
10
Delta temperature
hour


LAI = 5 layers = 10 chi.l = 1 lat = 42
Sun
Shade
1
2
3
4
5
6
7
8
9
10
Conductance (mmol/m2/s)
250
200
150
100
0 5 15
hour


LAI = 5 layers = 10 chi.l = 1 lat = 42
Sun
Shade
1
2
3
4
5
6
7
8
9
10
Relative Humidity
1.0
0.9
0.8
0.7
0 5 15
hour





Some important assumption of the multi-layer canopy model are

- it distributes the LAI equally among layers, this is not necessarily a

realistic assumption

- relative humidity increases with canopy depth which causes stomatal conductance to increase as well.
- by default photosynthetic parameters are constant in the profile but it is possible to make them depend on a profile of N concentration (see argument `lnControl`)

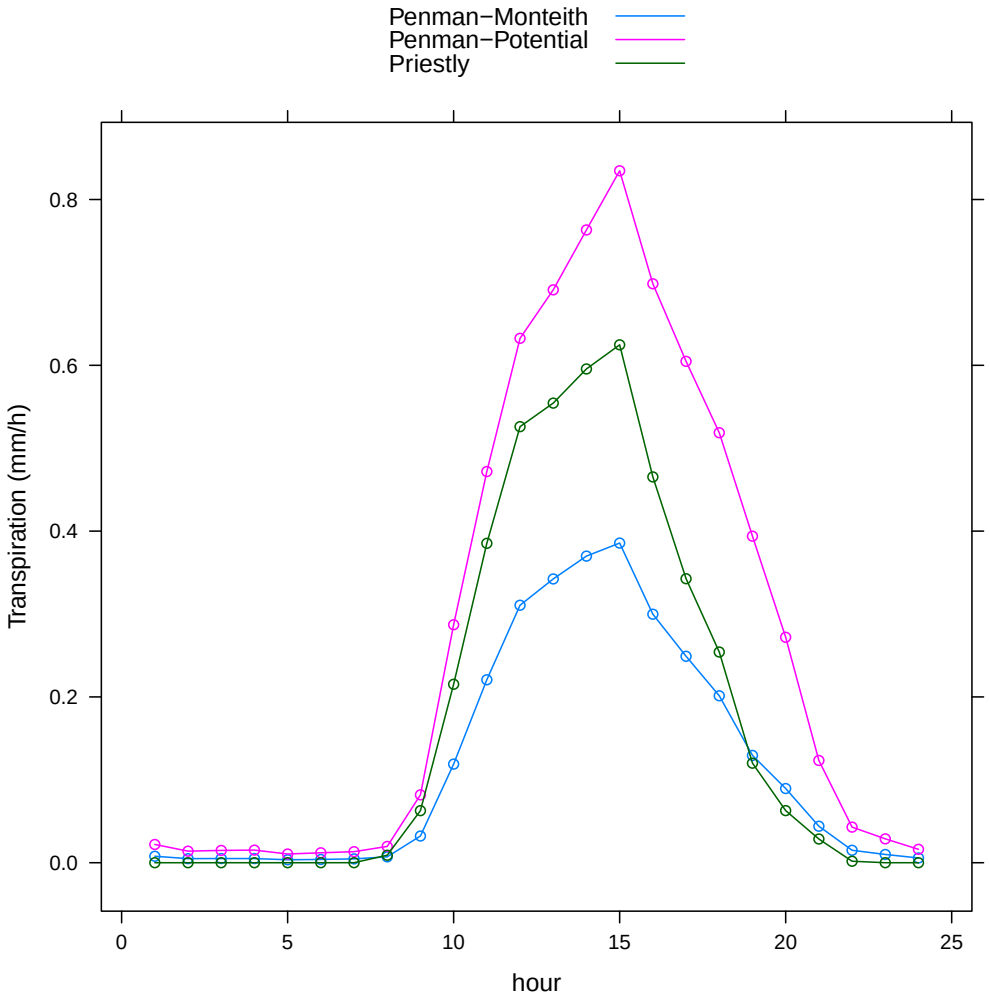
3.1 Parameters to adjust

Which parameters are relevant at the canopy level? Of course photosythetic parameters are important but these were discussed before so they are assume to be reasonable here. LAI is a very important input to this function so it is not really an adjustable parameter.

- `nlayers` The number of layers has an effect on many of the results. This can be modified by the user if there is a good rationale for doing it. It is possible that taller canopies benefit more from having multiple layers and shorter canopies benefit less.
- `kd` extinction coefficient for diffuse light. Although this can be calculated it is not at this point.
- `chi.l` is the ratio of horizontal to vertical projection of leaf area. Lower than 1 values for more erect canopies and less than 1 for canopies with higher proportion of flat leaves.
- `leafwidth` average leaf width.
- `heightFactor` factor relating LAI to height. Adjust it to match reasonable height for a crop.

3.2 Water: Calculation of Canopy Transpiration

`CanA` simulates transpiration using Priestly (driven by solar radiation and temperature), Penman-Potential (adjusted for the aerodynamic component) and Penman-Monteith (adjusted for the aerodynamic plus stomatal component).



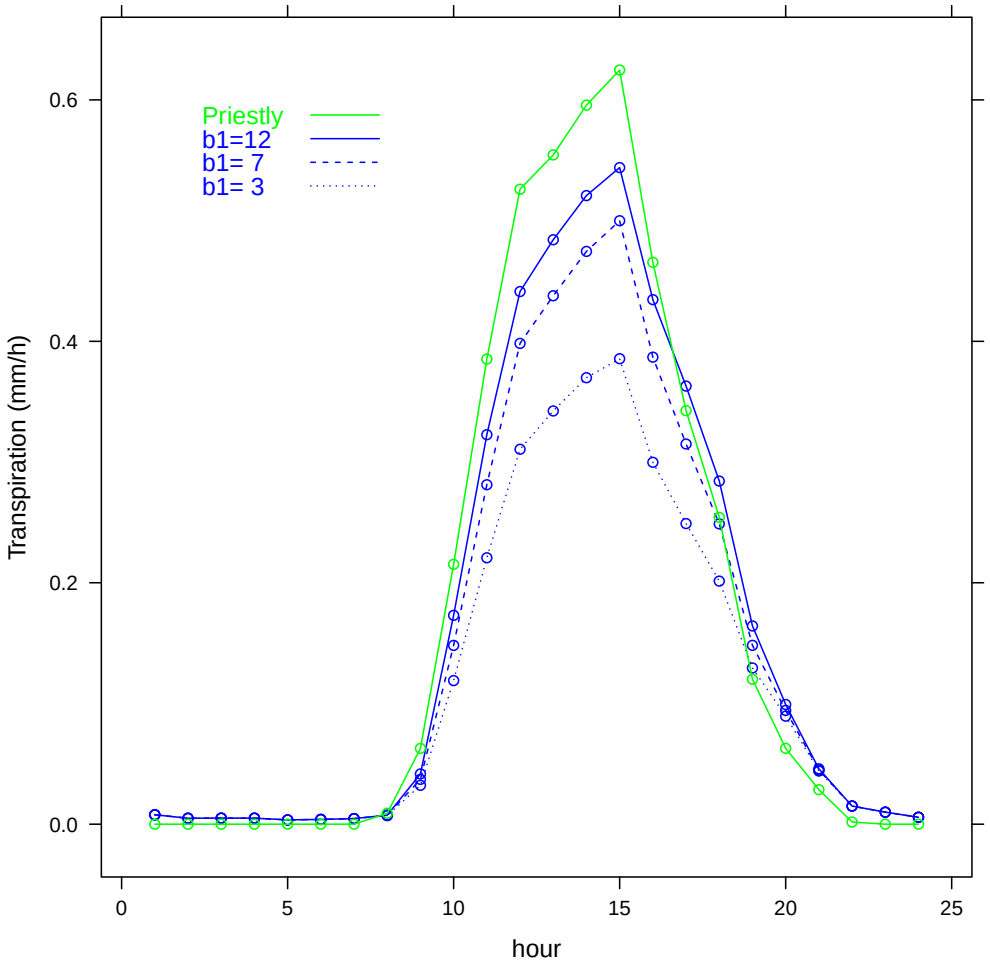

```
## CanopyTrans     TranPen  TranEpries
##    2.866192    6.582948    4.248740
```

At the moment Penman over estimates transpiration, Priestly and Penman-Monteith seem to be giving reasonable answers.

3.3 Water: Effect of Ball-Berry slope parameter

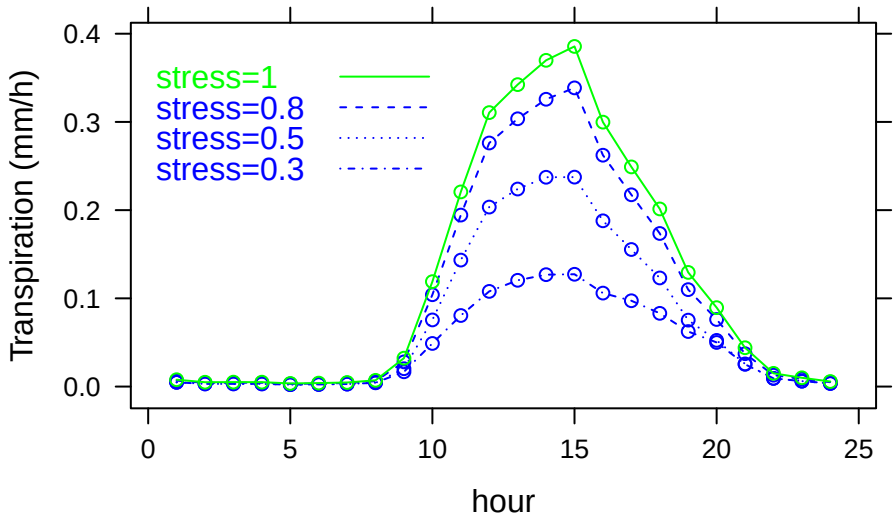
The slope of the Ball-Berry model can have a significant effect on the results, but only for the Penman-Monteith method. The parameters for Ball-Berry

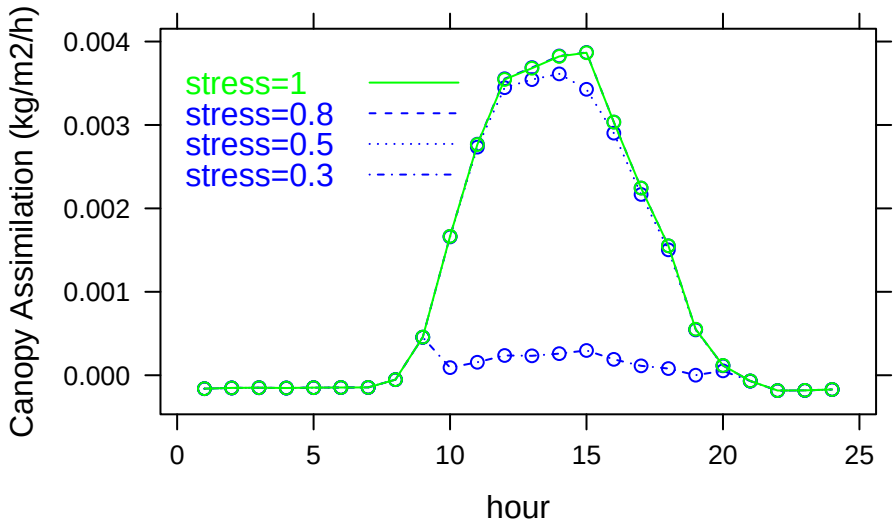
however should be set from previous literature data or from analysis of gas exchange measurements. The purpose of this is to show that it has a large effect. It is supplied by the `photoParms` function. This increases transpiration in the Penman-Monteith model but only up to a point. Priestly and Penman are almost always higher than Penman-Monteith.



3.4 Water: Effect of stress on diurnal transpiration

Another significant component that will affect transpiration during a day is the level of water stress the plant is experiencing.





TODO

- Include an example in which I see the effect of canopy height on diurnal transpiration.
- Include an example in which I see the effect of changing chi.l

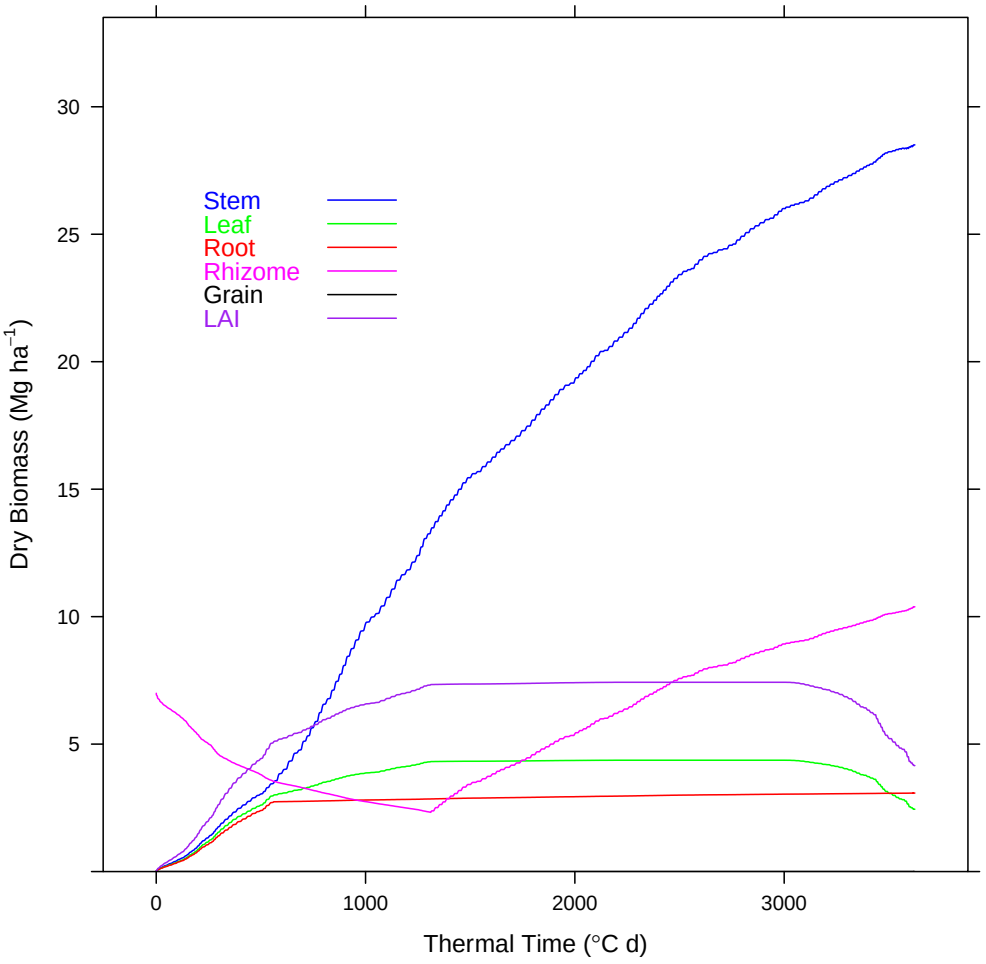
4 Carbon and Water: Biomass Crop Simulation

When the interest is to perform a simulation for a whole growing season, the function `BioGro` can be used. This function has as minimum input weather data for the whole year (365 days) at hourly time steps. The data can be generated using the `weach` function from daily data. One of the built-in datasets is `cmi04` which is weather data for Champaign, IL for 2004.

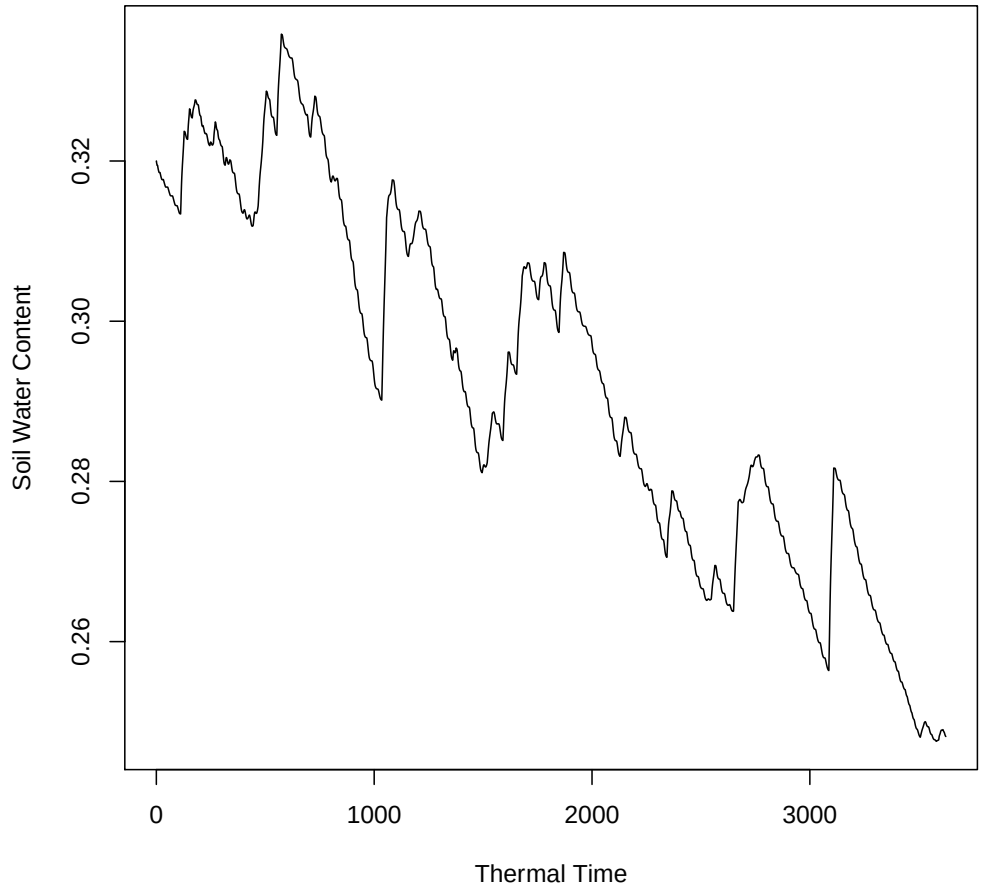
```
data(cmi04)
summary(cmi04)
```

```
##       year           doy             hour           SolarR
##  Min.   :2004   Min.   :  1.0   Min.   : 0.00   Min.   :   0.0
##  1st Qu.:2004   1st Qu.: 92.0   1st Qu.: 5.75   1st Qu.:   0.0
##  Median :2004   Median :183.5   Median :11.50   Median :   0.0
##  Mean   :2004   Mean   :183.5   Mean   :11.50   Mean   : 436.3
##  3rd Qu.:2004   3rd Qu.:275.0   3rd Qu.:17.25   3rd Qu.: 836.9
##  Max.   :2004   Max.   :366.0   Max.   :23.00   Max.   :2181.1
##       Temp              RH               WS
##  Min.   :-23.278   Min.   :0.2140   Min.   :0.2236
##  1st Qu.:  3.696   1st Qu.:0.6365   1st Qu.:1.2171
##  Median : 12.611   Median :0.7896   Median :1.9867
##  Mean   : 11.638   Mean   :0.7563   Mean   :2.3886
##  3rd Qu.: 20.278   3rd Qu.:0.9045   3rd Qu.:3.1306
##  Max.   : 32.444   Max.   :1.0000   Max.   :9.7942
##      precip
##  Min.   :0.00000
##  1st Qu.:0.00000
##  Median :0.00000
##  Mean   :0.11624
##  3rd Qu.:0.03175
##  Max.   :3.13267
```

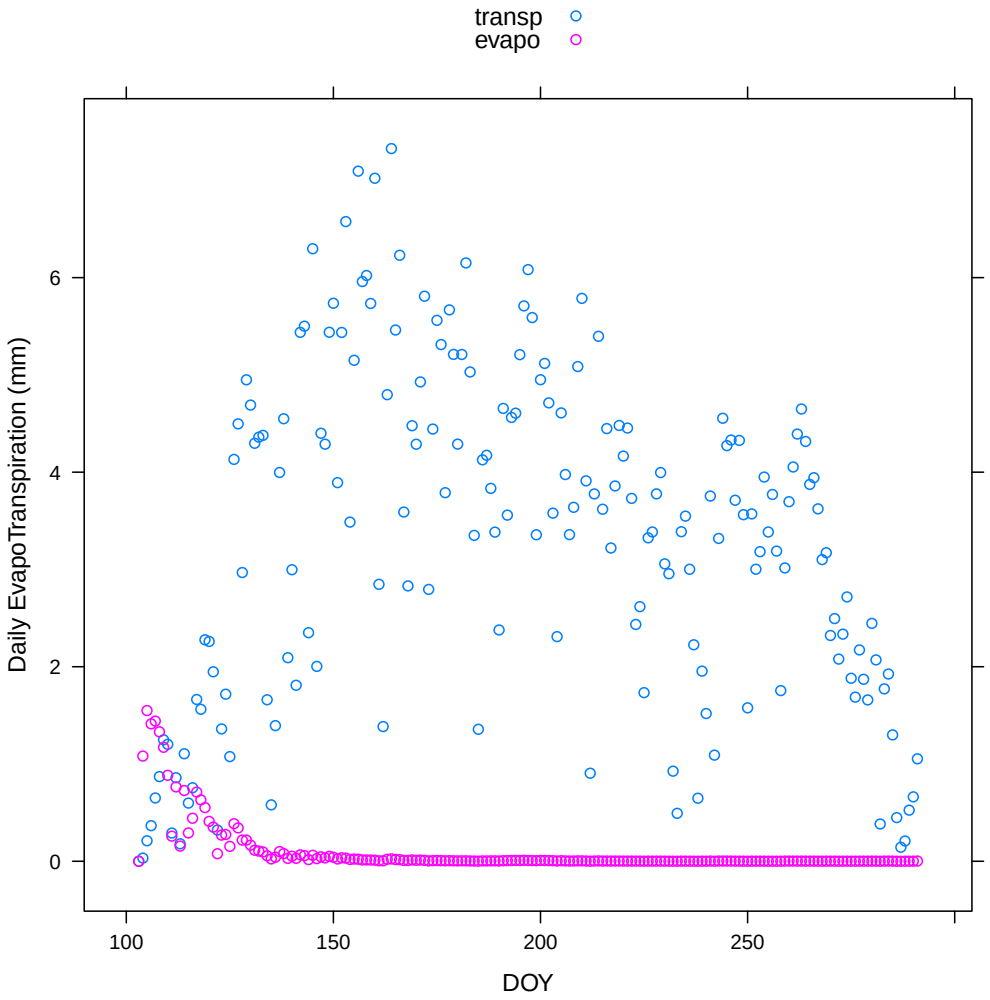
```
soilP <- soilParms(wsFun='linear')
res <- BioGro(cmi04, soilControl=soilP)
plot(res)
```



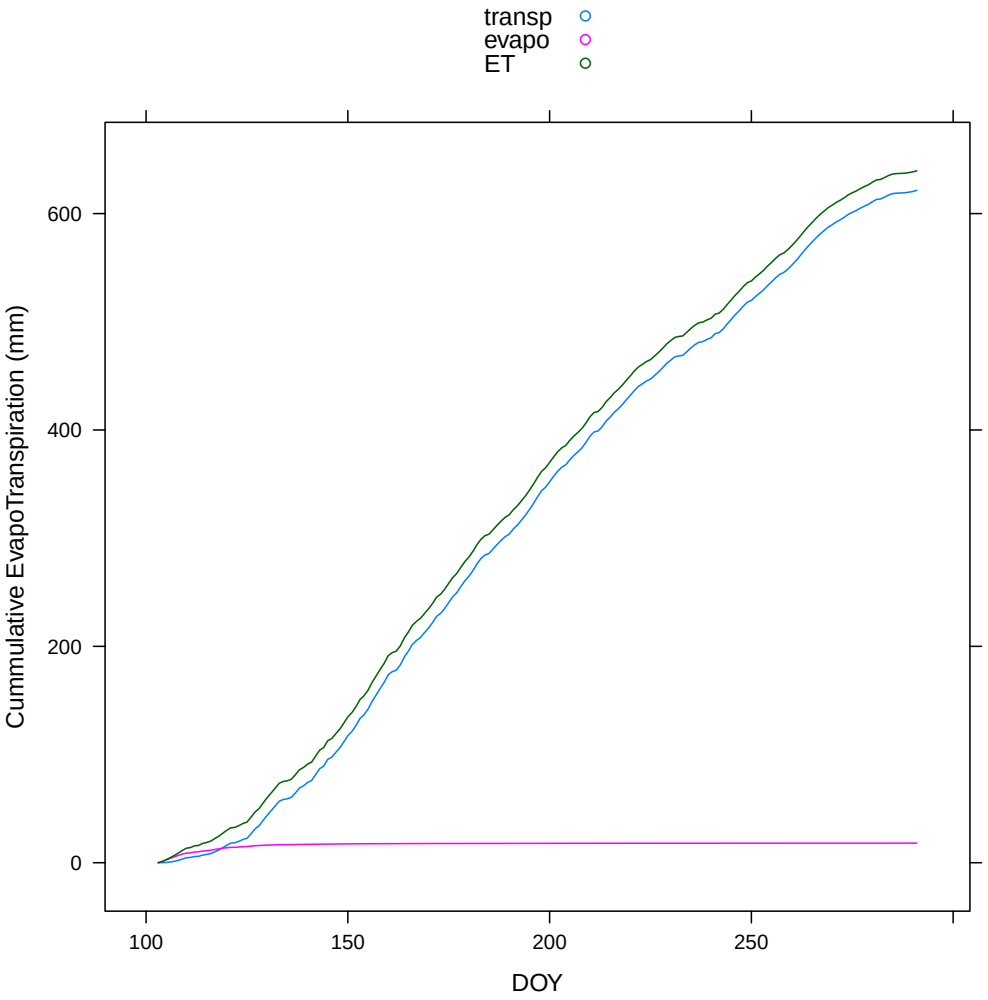

```
plot(res, plot.kind="SW")
```



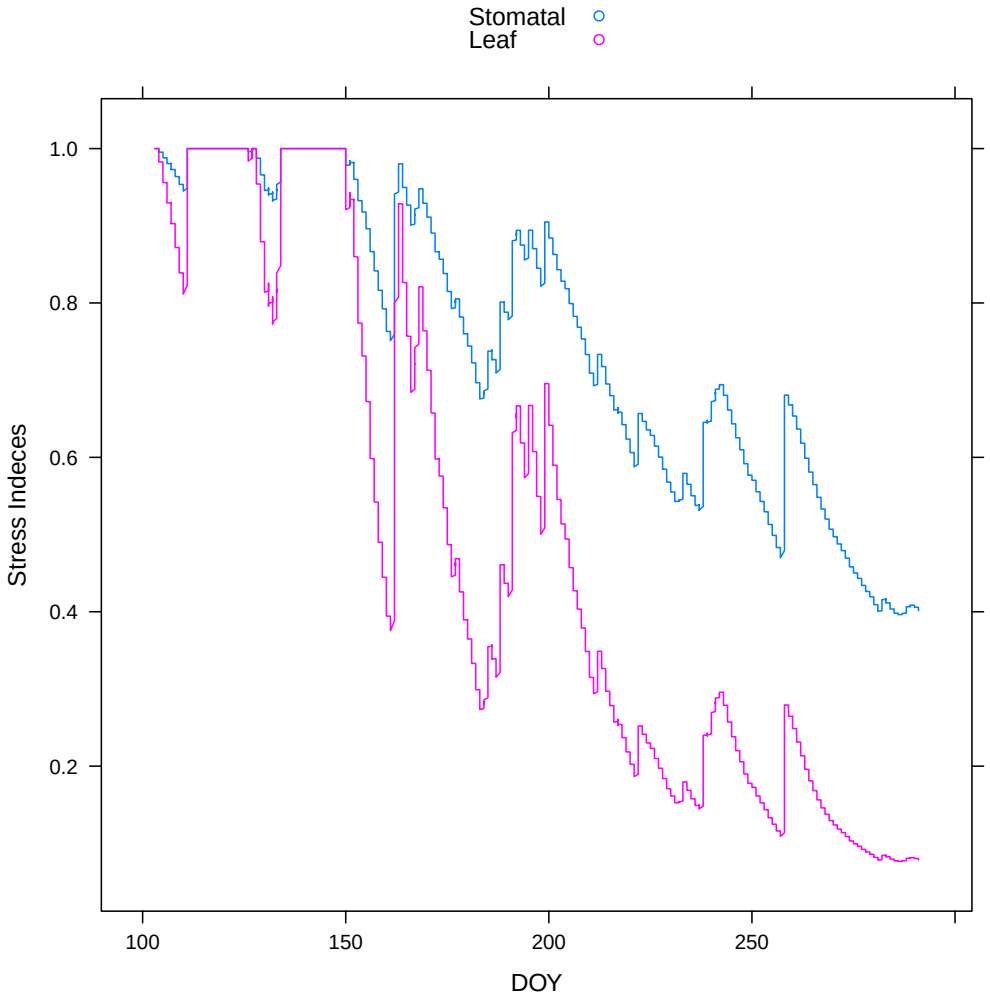

```
plot(res, plot.kind="ET")
```




```
plot(res, plot.kind="cumET")
```




```
plot(res, plot.kind="stress")
```




```
names(res)
```

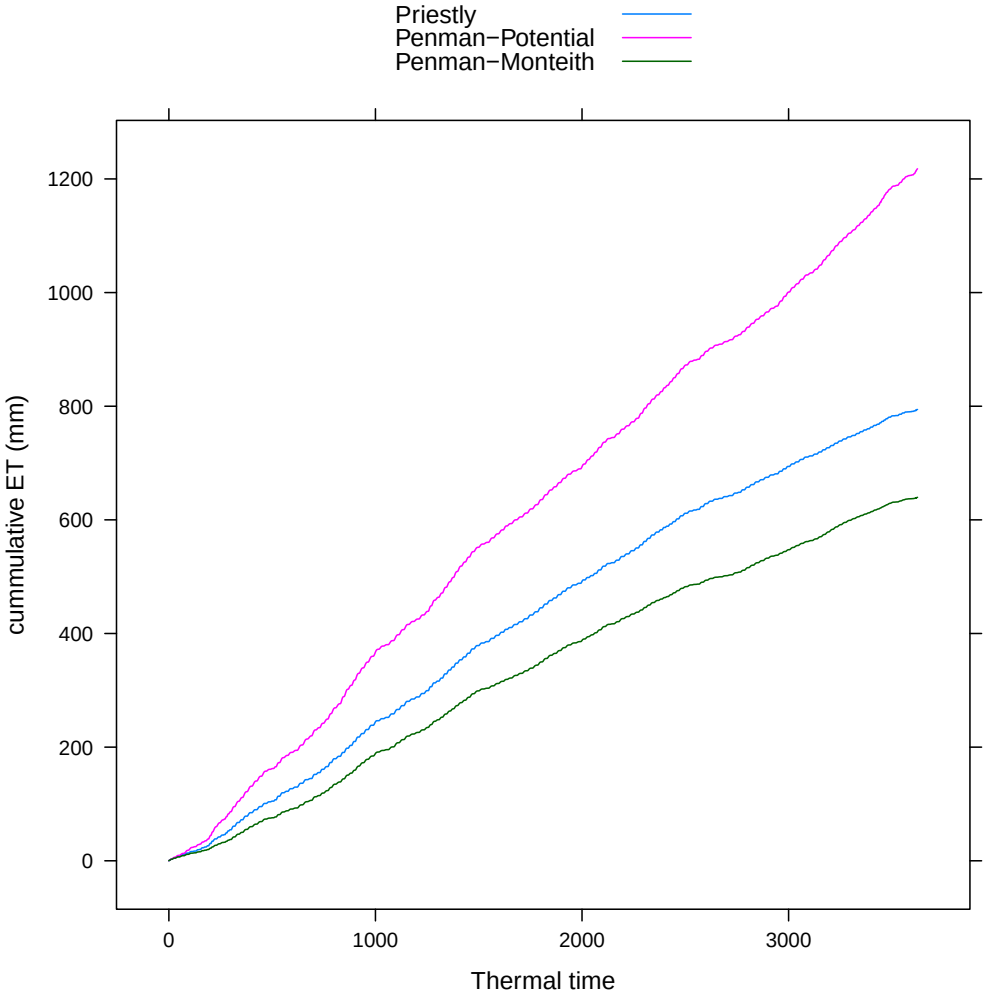
```
##  [1] "DayofYear"           "Hour"                "CanopyAssim"
##  [4] "CanopyTrans"         "Leaf"                "Stem"
##  [7] "Root"                "Rhizome"             "Grain"
## [10] "LAI"                 "ThermalT"            "SoilWatCont"
## [13] "StomatalCondCoefs"   "LeafReductionCoefs"  "LeafNitrogen"
## [16] "AboveLitter"         "BelowLitter"         "VmaxVec"
## [19] "AlphaVec"            "SpVec"               "MinNitroVec"
## [22] "RespVec"             "SoilEvaporation"     "cwsMat"
```

```
## [25] "psimMat"              "rdMat"              "SCpools"
## [28] "SNpools"              "LeafPsimVec"        "Drainage"
## [31] "Runoff"
```

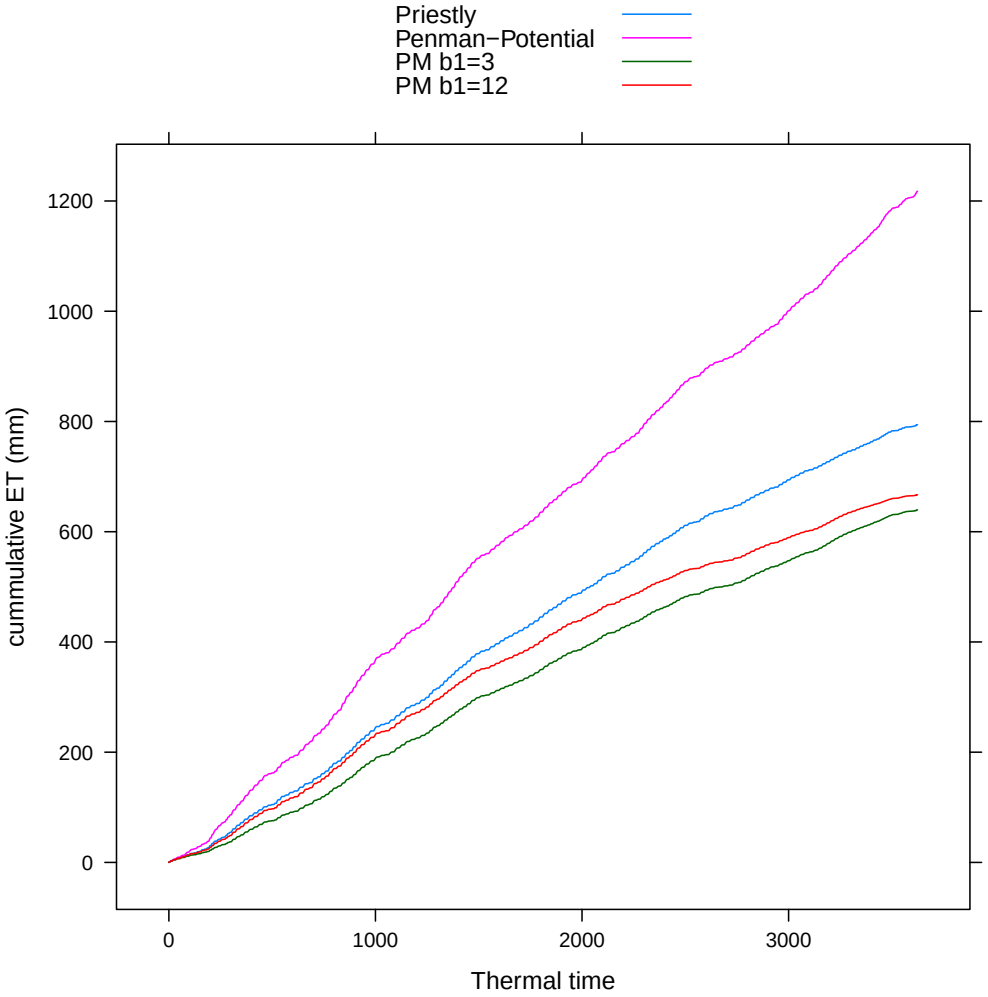
The last command `names(res)` shows the list of objects available for further manipulation.

4.1 Water: Calculation of EvapoTranspiration

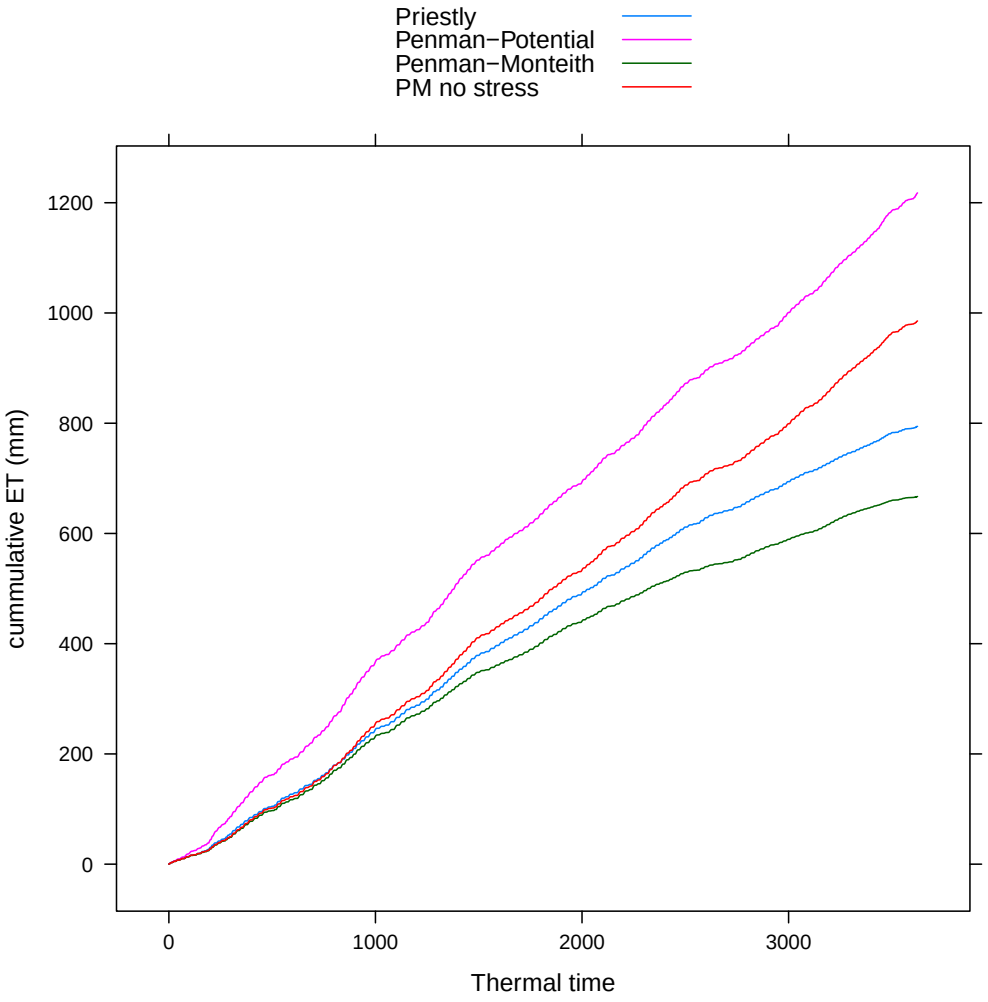
As in the `CanA` function transpiration can be calculated using Priestly, Penman-Potential or Penman-Monteith.



For Priestly and Penman-Potential the calculation of stomatal conductance does not affect the simulation, but for Penman-Monteith it does.



Penman-Moneith does not quite reach Priestly in this case because of stress. It is possible to perform simulations assuming that there is no stress but this will lead to results which approach Penman-Potential. When the no-stress option is selected the crop transpires freely unconstrained by soil water availability. This is not realistic but it is useful for testing and understanding transpiration processes.



4.2 Water: Balance for a growing season

The example below shows a water balance which consists of taking into account precipitation, evapotranspiration, drainage, runoff and change in water storage.

$$P - (ET + RO + DR + \Delta\Theta) = 0$$

where

P is precipitation (mm) ET is evapotranspiration (mm) RO is runoff (mm) DR is drainage (mm) $\Delta\Theta$ is change in soil water storage.

```
## Simple water budget
## P - ET + RO + DR + DeltaTheta = 0
data(cmi04)
day1 <- 100
dayn <- 270
cmi04.s <- subset(cmi04, doy > 99 & doy < 271)
P <- sum(cmi04.s$precip) ## in mm
iwc <- 0.29
soildepth <- 2
soilP <- soilParms(iWatCont=iwc, soilDepth=soildepth, soilLayers=1)
res <- BioGro(cmi04, day1=100, dayn=270, soilControl = soilP)
et <- res$CanopyTrans + res$SoilEvaporation
ET <- sum(et) * (1/0.9982) *0.1
## in mm, 0.9982 accounts for density of water
RO <- sum(res$Runoff) ## in mm
DR <- sum(res$Drainage) ## in mm
iTheta <- iwc * soildepth
fswc <- res$SoilWatCont[length(res$SoilWatCont)]
fswc

## [1] 0.2677877

fTheta <- fswc * soildepth
DeltaTheta <- (fTheta - iTheta) * 1e3 ## from m to mm
cbind(P, ET, DeltaTheta, RO, DR)

##             P       ET DeltaTheta RO        DR
## [1,] 502.158 546.4655  -44.42454  0 0.1275804

P - (ET + DeltaTheta + RO + DR)

## [1] -0.01058333
```

In this example there is no runoff and little drainage. There is a small numerical error, but the result is extremely close to zero.

4.3 Carbon and Water: Soil properties and parameters

Given that the model has been adequately described at the leaf and canopy level, when doing a simulation for the whole growing season the soil information becomes highly relevant. The basic information is supplied through the `soilParms` function.

```
soilP <- soilParms()
names(soilP)

##  [1] "FieldC"     "WiltP"      "phi1"       "phi2"
##  [5] "soilDepth"  "iWatCont"   "soilType"   "soilLayers"
##  [9] "soilDepths" "wsFun"      "scsf"       "transpRes"
## [13] "leafPotTh"  "hydrDist"   "rfl"        "rsec"
## [17] "rsdf"       "smthresh"   "lrt"        "lrf"
## [21] "respcoef"
```

Some of the details are available in `?BioGro`. The first two are important as they are the field capacity `FieldC` and wilting point `WiltP` if they are not supplied they are obtained from a default soil given by `soilType`. To look at the standard soils see

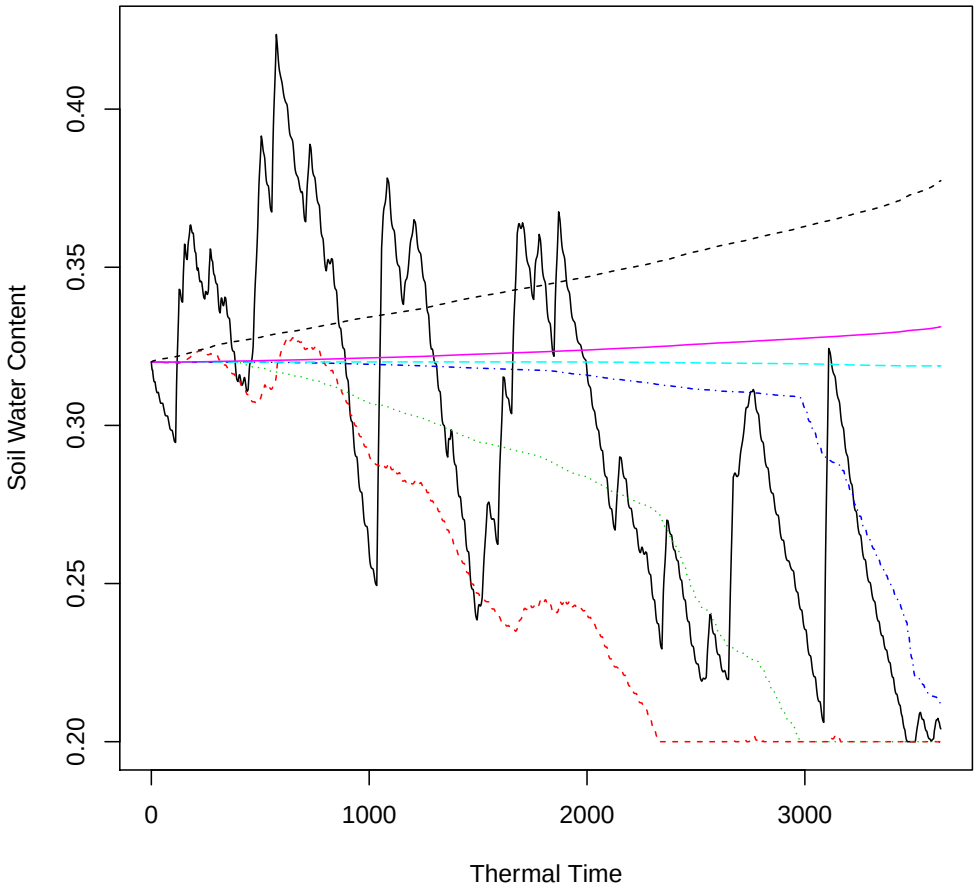
```
showSoilType(0)

## sand soil
## silt =  0.05
## clay =  0.03
## sand =  0.92
## air entry =  -0.7
## b =  1.7
## Ks =  0.0058
## satur =  0.4
## fieldc =  0.09
## wiltp =  0.03

showSoilType(5)
```

```
##  sandy clay loam
## silt =  0.13
## clay =  0.27
## sand =  0.6
## air entry =  -2.8
## b =  4
## Ks =  0.00012
## satur =  0.48
## fieldc =  0.26
## wiltp =  0.15

showSoilType(10)

##  clay
## silt =  0.2
## clay =  0.6
## sand =  0.2
## air entry =  -3.7
## b =  7.6
## Ks =  1.7e-05
## satur =  0.53
## fieldc =  0.4
## wiltp =  0.27
```

This shows a range of soils from clay (10) to sand (0) and an intermediate sandy clay loam.

Another important property is the soil depth. Typically crops have access to anywhere from 1 to 2.5 m of soil through their root exploration. If the number of layers of soil is equal to 1 then the soil is treated as a simple bucket and the crop roots have access to the entire profile. If the number of layers is larger than one the roots will only have access to the layers in which they have grown into. An example of a simulation using 7 layers and a soil depth of 3m and an empirical leaf reduction factor (`lrf=0.3e-3`). This type of empirical approach to the effect of water stress on senescence is used in other models such as APSIM.

```
soilP <- soilParms(soilLayers = 7, soilDepth = 3, lrf=0.3e-3)
res <- BioGro(cmi04, soilControl = soilP)
plot(res, plot.kind="SW")
```



The shallower layers are depleted while the deeper layers still have water in them. This also shows that as the top layer is depleted the crop takes up water from the layers just beneath it and then the next layer down and so on. All the layers started at the same level on day 1. This is the default behavior but it can be modified by changing the argument `iWatCont`. By default water moves from one layer to the next driven by soil water potential `hydrDist =`

TRUE this can be turned off but it will likely not produce reasonable results.

4.3.1 Soil Carbon: Century model

The modeling of soil carbon is based on the century model. In BioCro this can be simulated using the Century function (see ?Century). Some important first considerations for a given soil can be explored using the somc function which can be used to determine the amount of SOM in microbial, slow and passive pools. See the Century documentation https://www.nrel.colostate.edu/projects/century/MANUAL/html_manual/man96.html

```
somc()

## $soil.carbon.kg.m2
## [1] 3.9
##
## $soil.carbon.g.kg
## [1] 10
##
## $soil.carbon.Mg.ha
## [1] 39
##
## $SC6
## [1] 0.39
##
## $SC7
## [1] 7.41
##
## $SC8
## [1] 31.2
```

The main components of the soil organic matter are pools 6 (microbe), 7 (slow) and 8 (passive).

When conducting a simulation using BioGro the properties can be supplied using centuryParms.

```
centP <- centuryParms(om = 2.5, pp=c(0.02, 0.18, 0.8))
res <- BioGro(cmi04, centuryControl = centP)
```

```
names(res)
```

```
##  [1] "DayofYear"           "Hour"                "CanopyAssim"
##  [4] "CanopyTrans"         "Leaf"                "Stem"
##  [7] "Root"                "Rhizome"             "Grain"
## [10] "LAI"                 "ThermalT"            "SoilWatCont"
## [13] "StomatalCondCoefs"   "LeafReductionCoefs"  "LeafNitrogen"
## [16] "AboveLitter"         "BelowLitter"         "VmaxVec"
## [19] "AlphaVec"            "SpVec"               "MinNitroVec"
## [22] "RespVec"             "SoilEvaporation"     "cwsMat"
## [25] "psimMat"             "rdMat"               "SCpools"
## [28] "SNpools"             "LeafPsimVec"         "Drainage"
## [31] "Runoff"
```

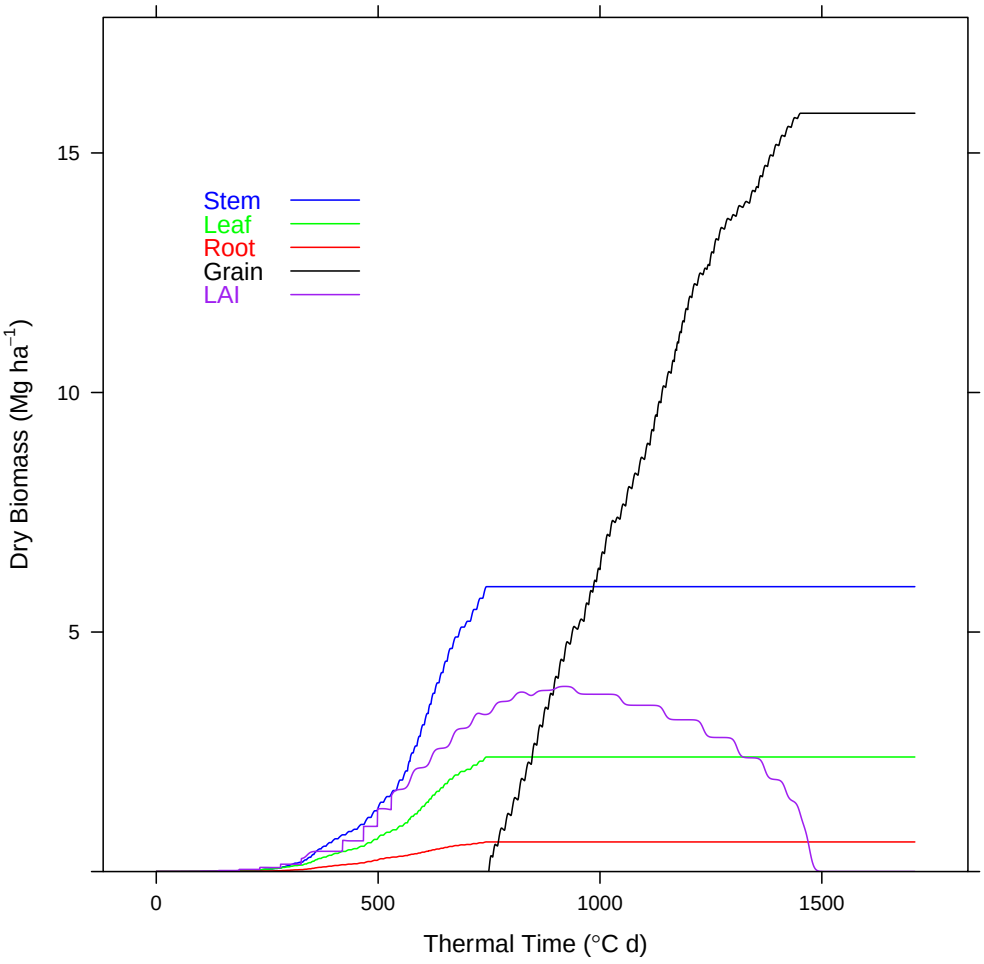
The vector `RespVec` represents the soil microbial respiration. `SCpools` represents the soil carbon pools (some are actually surface carbon pools such as litter. Similarly `SNpools` are the corresponding nitrogen pools.

```
## data(EngWea94i) ## Load weather data
## data(annualDB) ## Load biomass data
## seneP <- seneParms(senLeaf = 1800)
## photoP <- photoParms(alpha=0.05)
## mxg <- BioGro(EngWea94i, day1=131, dayn=320,
##               seneControl = seneP,
##               photoControl = photoP)
## plot(mxg, annualDB, ylim=c(0,30))
```

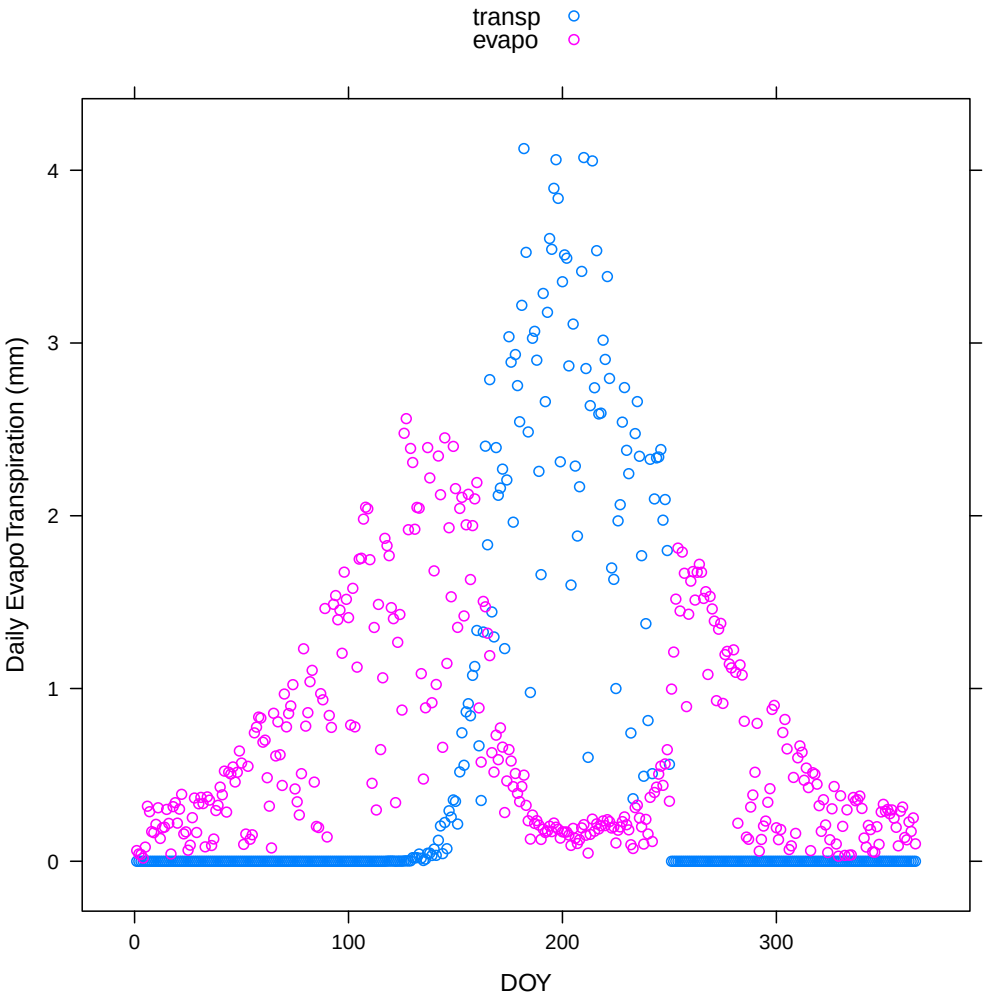
5 Maize

A maize model is under development.

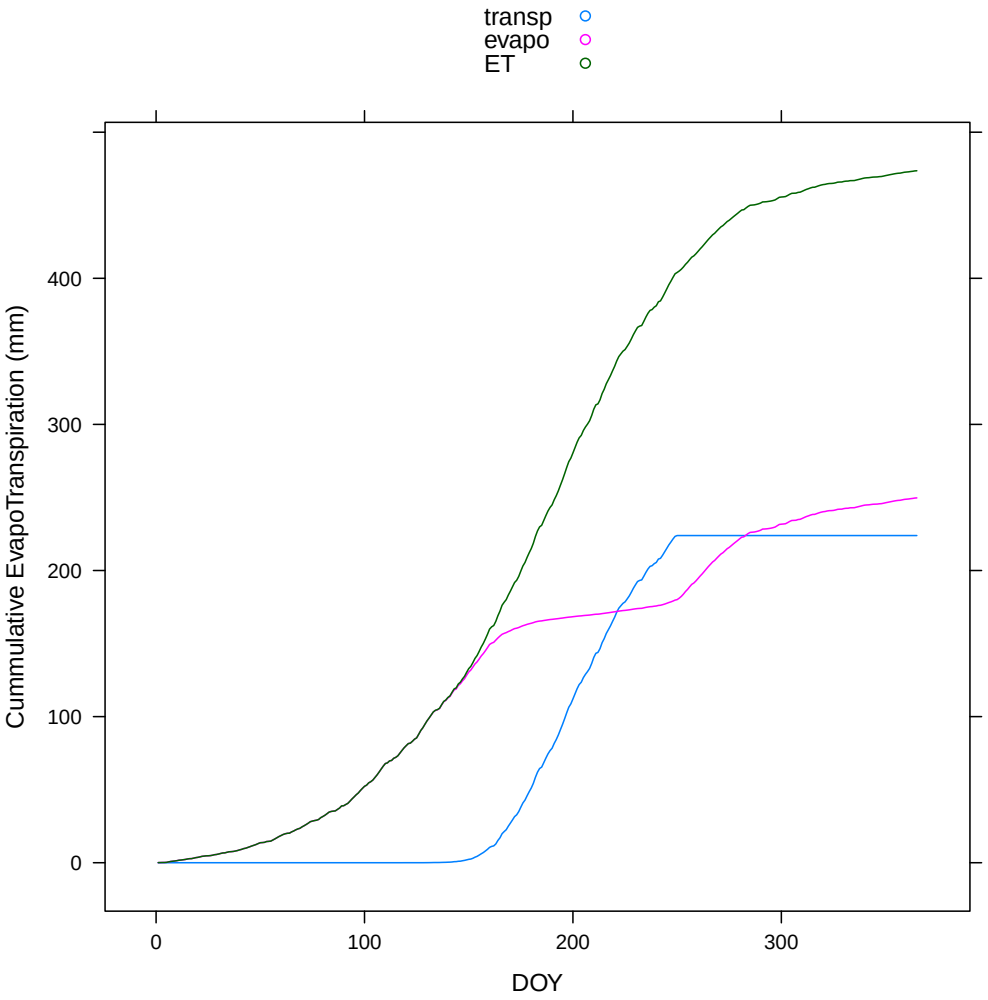
```
data(cmi04)
soilP <- soilParms(soilDepth = 2)
res <- MaizeGro(cmi04, plant.day = 110, emerge.day = 117, harvest.day = 280,
    soilControl = soilP)
plot(res)
```



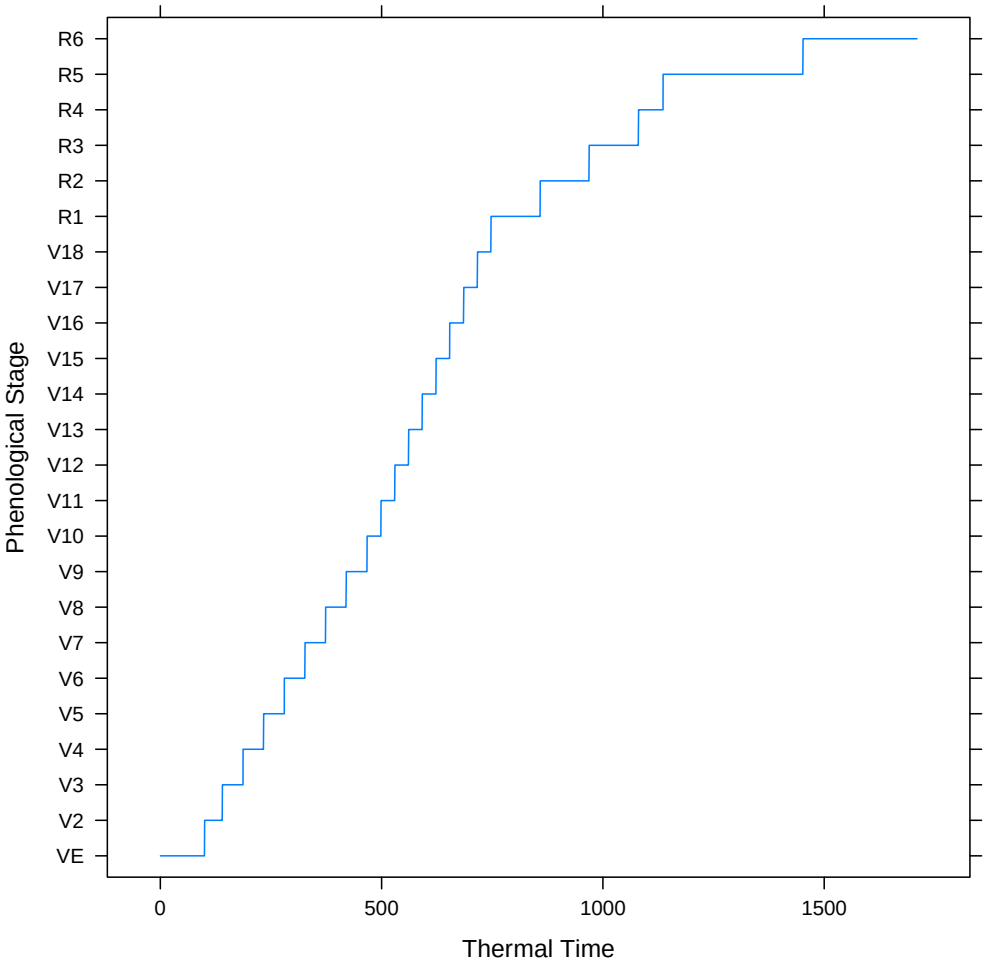

```
plot(res, plot.kind = "ET")
```



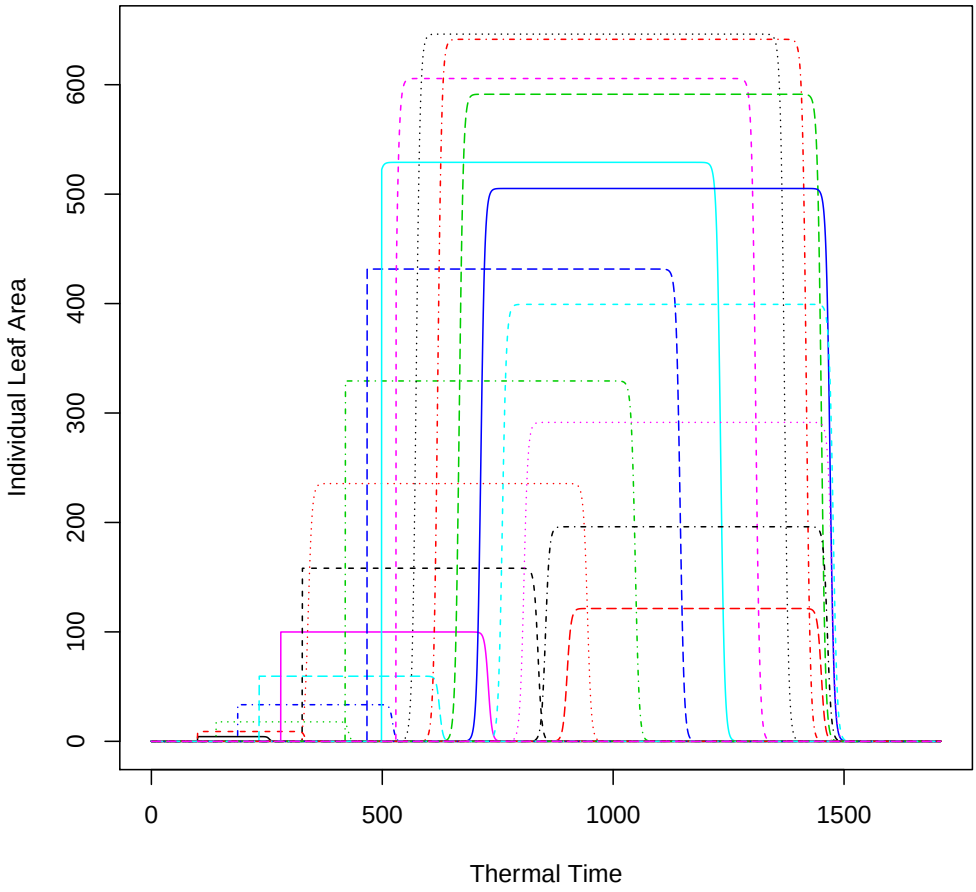

```
plot(res, plot.kind = "cumET")
```




```
plot(res, plot.kind = "pheno")
```




```
plot(res, plot.kind = "LAI")
```


Individual Leaf Area
0
100
200
300
400
500
600
0
500
1000
1500
Thermal Time