Exercise 2 solution

Miquel De Cáceres

2024-11-07

Exercise setting

Overall goal

Learn how to use *medfate* for forest water balance simulations.

Specific objectives

1. Perform a basic water balance run on a real-case data and inspect the results
2. Evaluate the performance of the water balance model with observed data
3. Perform an advanced water balance run on the same data and inspect the results
4. Compare the results and performance between the two models

Exercise material

- Exercise_1.Rmd
- fontblanche.rds

Font-Blanche research forest

- The Font-Blanche research forest is located in southeastern France (43º14'27'' N 5°40'45'' E) at 420 m elevation)
- The stand is composed of a top strata of *Pinus halepensis* (Aleppo pine) reaching about 12 m, a lower strata of *Quercus ilex* (holm oak), reaching about 6 m, and an understorey strata dominated by *Quercus coccifera* and *Phillyrea latifolia*.
- Soils are shallow and rocky, have a low retention capacity and are of Jurassic limestone origin.
- The climate is Mediterranean, with a water stress period in summer, cold or mild winters and most precipitation occurring between September and May.

Target stand

- The experimental site, which is dedicated to study forest carbon and water cycles, has an enclosed area of 80×80 m but our target stand is a quadrat of dimensions 25×25 m.
- The following observations are available for year 2014:
  - Stand total evapotranspiration estimated using an Eddy-covariance flux tower.
  - Soil moisture content of the topmost (0-30 cm) layer.

- Transpiration estimates per leaf area, derived from sapflow measurements for *Q. ilex* and *P. halepensis*.
- Pre-dawn and midday leaf water potentials for *Q. ilex* and *P. halepensis*.

Exercise solution

Step 1. Load Font-Blanche data

We are given all the necessary data, bundled in a single list:

```
fb <- readRDS("../StudentRdata/fb_data.rds")
names(fb)
```

```
##  [1] "siteData"       "treeData"       "shrubData"      "customParams"
##  [5] "measuredData"   "meteoData"      "miscData"       "soilData"
##  [9] "terrainData"    "remarks"        "sp_params"      "forest_object1"
```

Step 2. Build forest object

Element `fb$treeData` is already in the appropriate format for `forest` objects. Therefore, we can easily plug the tree data into a forest object:

```
fb_forest <- emptyforest()
fb_forest$treeData <- fb$treeData
```

and examine its characteristics:

```
summary(fb_forest, SpParamsMED)
```

```
## Tree BA (m2/ha): 24.4861797  adult trees: 23.8297536  saplings: 0.6564261
## Density (ind/ha) adult trees: 3360  saplings: 1248  shrubs (estimated): 0
## Cover (%) adult trees: 100  saplings: 27.9158707  shrubs: 0  herbs: 0
## LAI (m2/m2) total: 2.7  adult trees: 2.4418971  saplings: 0.2581029  shrubs: 0  herbs: 0
## Fuel loading (kg/m2) total: 0.8739882  adult trees: 0.7934566  saplings: 0.0805316  shrubs: 0  herbs
## PAR ground (%): NA  SWR ground (%): NA
```

Note, incidentally, that LAI estimates come from the sum of column `LAI`, without any use of allometric relationships.

Step 3. Initialize soil object

A data frame with soil physical attributes are defined in:

```
fb$soilData
```

```
##   widths clay sand om   bd rfc
## 1    300   39   26  6 1.45  50
## 2    700   39   26  3 1.45  65
## 3   1000   39   26  1 1.45  90
## 4   2500   39   26  1 1.45  95
```

We need, however, to initialize a `soil` object to complete the hydraulic parameters: .code80[

```
fb_soil <- soil(fb$soilData)
```

From which we can estimate the water holding capacity and extractable water capacity for each layer (in mm) using functions `soil_waterFC()` and `soil_waterExtractable()`:

```
soil_waterFC(fb_soil)
```

```
## [1] 58.32111 92.92727 37.30063 46.62578
```

```
soil_waterExtractable(fb_soil)
```

```
## [1] 26.06443 41.96683 16.97066 21.21332
```

So the sums would be:

```
sum(soil_waterFC(fb_soil))
```

```
## [1] 235.1748
```

```
sum(soil_waterExtractable(fb_soil))
```

```
## [1] 106.2152
```

The same information can also be found in the output of `summary()`.

```
summary(fb_soil)
```

```
## Soil depth (mm): 4500
##
## Layer  1  [ 0  to  300 mm ]
##     clay (%): 39 silt (%): 29 sand (%): 26 organic matter (%): 6 [ Clay loam ]
##     Rock fragment content (%): 50 Macroporosity (%): 7
##     Theta WP (%): 25 Theta FC (%): 39 Theta SAT (%): 54 Theta current (%) 39
##     Vol. WP (mm): 37 Vol. FC (mm): 58 Vol. SAT (mm): 81 Vol. current (mm): 58
##     Temperature (Celsius): NA
##
## Layer  2  [ 300  to  1000 mm ]
##     clay (%): 39 silt (%): 32 sand (%): 26 organic matter (%): 3 [ Clay loam ]
##     Rock fragment content (%): 65 Macroporosity (%): 7
##     Theta WP (%): 24 Theta FC (%): 38 Theta SAT (%): 49 Theta current (%) 38
##     Vol. WP (mm): 59 Vol. FC (mm): 93 Vol. SAT (mm): 121 Vol. current (mm): 93
##     Temperature (Celsius): NA
##
## Layer  3  [ 1000  to  2000 mm ]
##     clay (%): 39 silt (%): 34 sand (%): 26 organic matter (%): 1 [ Clay loam ]
##     Rock fragment content (%): 90 Macroporosity (%): 7
##     Theta WP (%): 24 Theta FC (%): 37 Theta SAT (%): 47 Theta current (%) 37
##     Vol. WP (mm): 24 Vol. FC (mm): 37 Vol. SAT (mm): 47 Vol. current (mm): 37
```

```
##     Temperature (Celsius): NA
##
## Layer  4  [ 2000  to  4500 mm ]
##     clay (%): 39 silt (%): 34 sand (%): 26 organic matter (%): 1 [ Clay loam ]
##     Rock fragment content (%): 95 Macroporosity (%): 7
##     Theta WP (%): 24 Theta FC (%): 37 Theta SAT (%): 47 Theta current (%) 37
##     Vol. WP (mm): 29 Vol. FC (mm): 47 Vol. SAT (mm): 58 Vol. current (mm): 47
##     Temperature (Celsius): NA
##
## Total soil saturated capacity (mm): 306
## Total soil water holding capacity (mm): 235
## Total soil extractable water (mm): 106
## Total soil current Volume (mm): 235
## Saturated water depth (mm): NA
```

Step 4. Species parameters

We will normally take `SpParamsMED` as starting point for species parameters:

```
data("SpParamsMED")
```

However, sometimes one may wish to override species defaults with custom values. In the case of FontBlanche there is a table of preferred values for some parameters:

```
fb$customParams
```

```
##              Species VCleaf_P12 VCleaf_P50 VCleaf_P88 VCleaf_slope VCstem_P12
## 1 Phillyrea latifolia         NA         NA         NA           NA  -1.971750
## 2    Pinus halepensis         NA         NA         NA           NA  -3.707158
## 3        Quercus ilex         NA         NA         NA           NA  -4.739642
##   VCstem_P50 VCstem_P88 VCstem_slope VCroot_P12 VCroot_P50 VCroot_P88
## 1      -6.50 -11.028250           11         NA         NA         NA
## 2      -4.79  -5.872842           46         -1  -1.741565  -2.301482
## 3      -6.40  -8.060358           30         NA         NA         NA
##   VCroot_slope VCleaf_kmax LeafEPS LeafPI0 LeafAF StemEPS StemPI0 StemAF Gswmin
## 1           NA        3.00   12.38   -2.13    0.5   12.38   -2.13    0.4  0.002
## 2           NA        4.00    5.31   -1.50    0.6    5.00   -1.65    0.4  0.001
## 3           NA        2.63   15.00   -2.50    0.4   15.00   -2.50    0.4  0.002
##   Gswmax    Gs_P50 Gs_slope    Al2As
## 1 0.2200 -2.207094 89.41176       NA
## 2 0.2175 -1.871216 97.43590  631.000
## 3 0.2200 -2.114188 44.70588 1540.671
```

We can use function `modifySpParams()` to replace the values of parameters for the desired traits, leaving the rest unaltered:

```
fb_SpParams <- modifySpParams(SpParamsMED, fb$customParams)
```

Steps 5-6. Basic water balance

Since we are about to run a basic water balance simulation, we initialize a simulation control parameter list with `transpirationMode = "Granier"`, i.e.:

```
fb_control <- defaultControl("Granier")
```

and we assemble our inputs into a `spwbInput` object, using:

```
fb_x1 <- spwbInput(fb_forest, fb_soil, fb_SpParams, fb_control)
```

The daily weather data comprises one year:

```
fb_meteo <- fb$meteoData
nrow(fb_meteo)
```

```
## [1] 365
```

Now, we are ready to launch the simulation:

```
fb_basic <- spwb(fb_x1, fb_meteo,  elevation = 420, latitude = 43.24083)
```

```
## Package 'meteoland' [ver. 2.2.2]

## Initial plant water content (mm): 17.1947
## Initial soil water content (mm): 213.886
## Initial snowpack content (mm): 0
## Performing daily simulations
##
##  [Year 2014]:............
##
## Final plant water content (mm): 17.1952
## Final soil water content (mm): 236.416
## Final snowpack content (mm): 0
## Change in plant water content (mm): 0.000480745
## Plant water balance result (mm): 0.000480745
## Change in soil water content (mm): 22.5304
## Soil water balance result (mm): 22.5304
## Change in snowpack water content (mm): 0
## Snowpack water balance result (mm): 0
## Water balance components:
##   Precipitation (mm) 1066 Rain (mm) 1066 Snow (mm) 0
##   Interception (mm) 141 Net rainfall (mm) 925
##   Infiltration (mm) 827 Infiltration excess (mm) 98 Saturation excess (mm) 268 Capillarity rise (mm)
##   Soil evaporation (mm) 26  Herbaceous transpiration (mm) 0 Woody plant transpiration (mm) 313
##   Plant extraction from soil (mm) 313  Plant water balance (mm) 0 Hydraulic redistribution (mm) 7
##   Runoff (mm) 365 Deep drainage (mm) 198
```

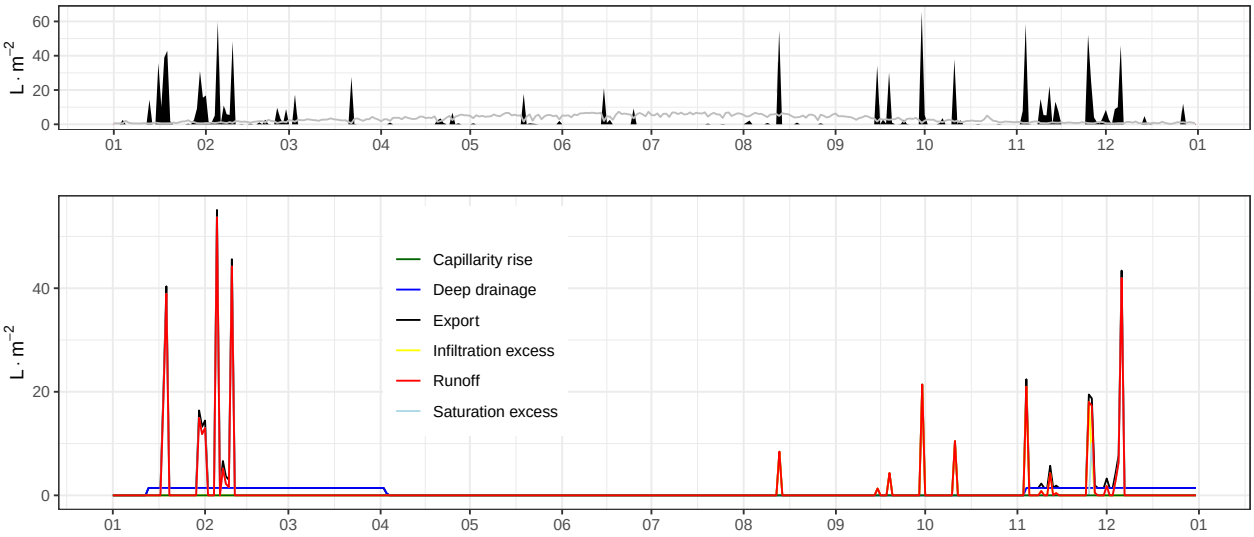
Step 7. Examine precipitation events, runoff and deep drainage

Surface run-off occurs the same day as precipitation events, whereas deep drainage can last for some days after the event:

```
g1<-plot(fb_basic)+scale_x_date(date_breaks = "1 month", date_labels = "%m")+
  theme(legend.position = "none")
g2<-plot(fb_basic, "Export")+scale_x_date(date_breaks = "1 month", date_labels = "%m")+
  theme(legend.position = c(0.35,0.60))
```

```
## Warning: A numeric ‘legend.position‘ argument in ‘theme()‘ was deprecated in ggplot2
## 3.5.0.
## i Please use the ‘legend.position.inside‘ argument of ‘theme()‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

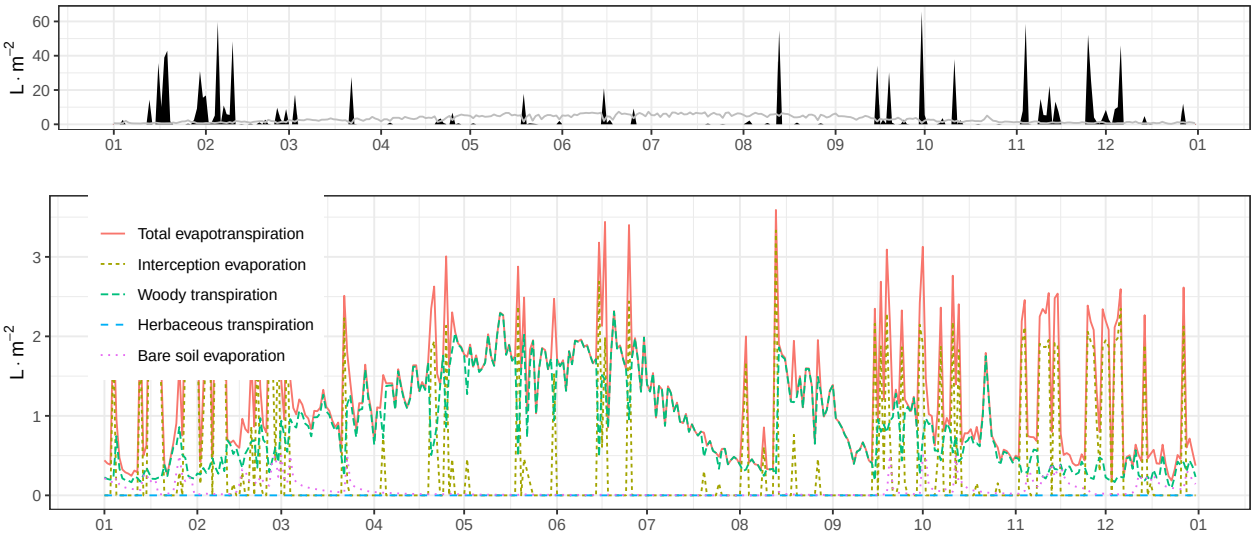
```
plot_grid(g1,g2, ncol=1, rel_heights = c(0.5,1))
```



Step 8. Examine evapotranspiration flows

Precipitation events also generate flows of intercepted water the same day of the event. Evaporation from the bare soil can proceed some days after the event. Transpiration flow is the dominant one in most days, decreasing in summer due to drought.

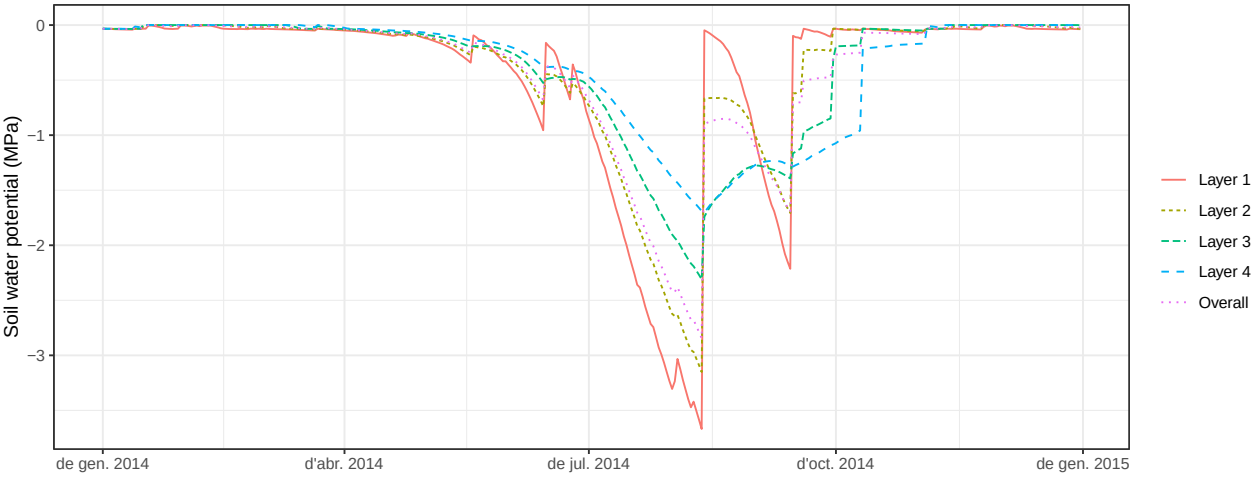
```
g1<-plot(fb_basic)+scale_x_date(date_breaks = "1 month", date_labels = "%m")+
  theme(legend.position = "none")
g2<-plot(fb_basic, "Evapotranspiration")+scale_x_date(date_breaks = "1 month", date_labels = "%m")+
  theme(legend.position = c(0.13,0.73))
plot_grid(g1,g2, ncol=1, rel_heights = c(0.5,1))
```



Step 9. Soil water potential dynamics

We can display the dynamics of water potential in different soil layers using:

```
plot(fb_basic, "SoilPsi")
```



Tip: Normally, we should expect lower layers to have a less dynamic behaviour, but strange results can occur if, for instance, a large proportion of roots is in deeper layers.

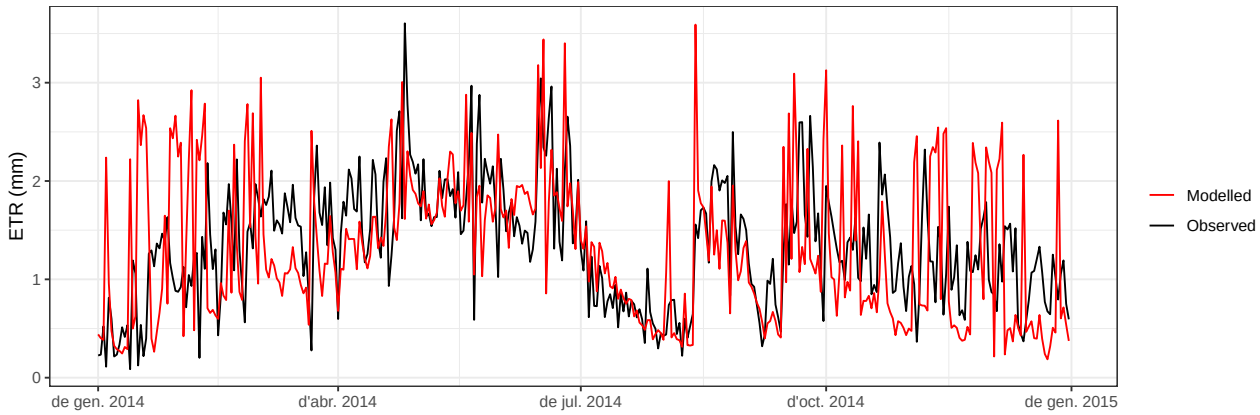
Steps 10-12. Evaluation of stand evapotranspiration

Observations are in element `measuredData` of the list:

```
fb_observed <- fb$measuredData
```

We can compare the observed vs modelled total evapotranspiration by plotting the two time series:

```
evaluation_plot(fb_basic, fb_observed, type = "ETR", plotType="dynamics")
```



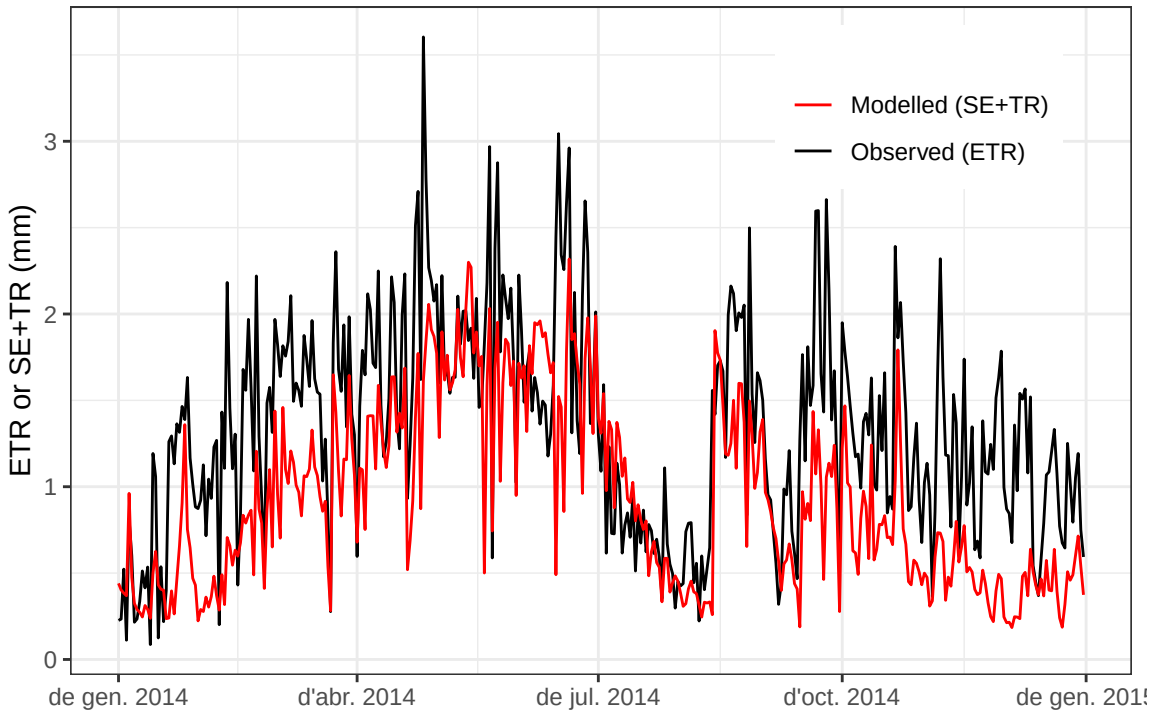
It is easy to see that in rainy days the predicted evapotranspiration is much higher than that of the observed data.

Steps 10-12. Evaluation of stand evapotranspiration

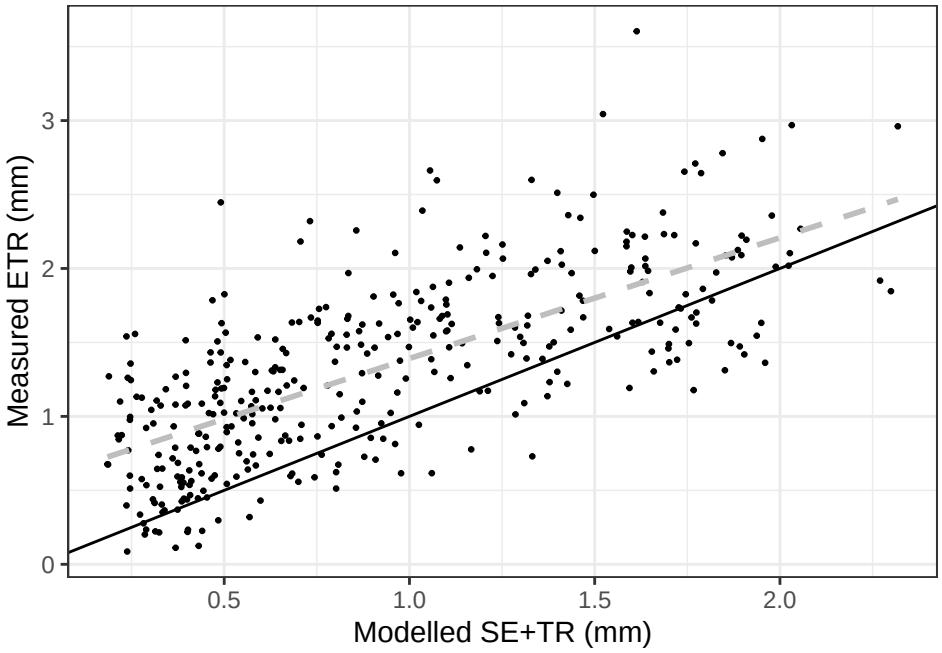
We repeat the comparison but excluding the intercepted water from modeled results:

Time series plot

```
evaluation_plot(fb_basic, fb_observed,
    type = "SE+TR", plotType="dynamics")+
  theme(legend.position = c(0.8,0.85))
```



```
evaluation_plot(fb_basic, fb_observed,
    type = "SE+TR", plotType="scatter")
```



Where we see a reasonably good relationship, but the model tends to underestimate total evapotranspiration during seasons with low evaporative demand.

Steps 10-12. Evaluation of stand evapotranspiration

Function `evaluation_stats()` allows us to generate evaluation statistics:

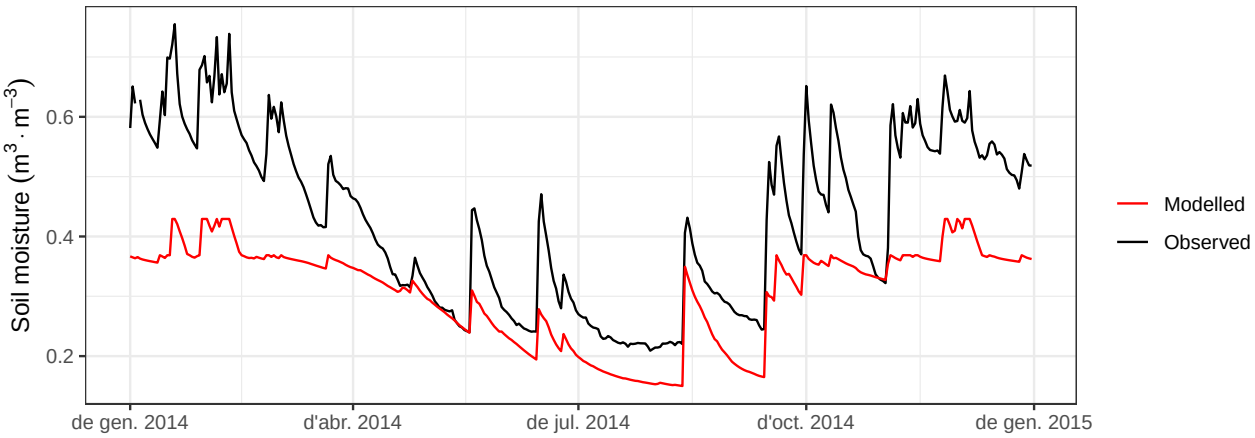
```
evaluation_stats(fb_basic, fb_observed, type = "SE+TR")
```

```
##            n         Bias     Bias.rel          MAE      MAE.rel            r
## 365.00000000  -0.40453685 -30.36201568   0.48384513  36.31440058   0.70498484
##          NSE      NSE.abs
##   0.03907060   0.02826233
```

Step 13. Evaluation of soil moisture content

We can now compare the soil moisture content dynamics using:

```
evaluation_plot(fb_basic, fb_observed, type = "SWC", plotType="dynamics")
```



The two series have similar shape but not absolute values. This may be an indication that the parameters of the soil water retention curve do not match the data produced by the moisture sensor.

We can calculate evaluation statistics using:

```
evaluation_stats(fb_basic, fb_observed, type = "SWC")
```

```
##            n         Bias     Bias.rel          MAE      MAE.rel            r
## 364.00000000  -0.12212079 -28.47040214   0.12220446  28.48990865   0.91255597
##          NSE      NSE.abs
##  -0.01186163   0.04257990
```

Note that, despite the bias, the correlation is rather high.

Step 14. Advanced water/energy balance

Since we are about to run a advanced water balance simulation, we initialize a simulation control parameter list with `transpirationMode = "Sperry"`, i.e.:

```
fb_control <- defaultControl("Sperry")
```

and assemble our inputs into a `spwbInput` object, using:

```
fb_x2 <- spwbInput(fb_forest, fb_soil, fb_SpParams, fb_control)
```

Finally, we launch the simulation (takes ~ 8 seconds):

```
fb_adv <- spwb(fb_x2, fb_meteo,  elevation = 420, latitude = 43.24083)
```

```
## Initial plant water content (mm): 31.8864
## Initial soil water content (mm): 213.886
## Initial snowpack content (mm): 0
## Performing daily simulations
##
##  [Year 2014]:............
##
## Final plant water content (mm): 31.4636
```

```
## Final soil water content (mm): 235.068
## Final snowpack content (mm): 0
## Change in plant water content (mm): -0.422751
## Plant water balance result (mm): -1.97044e-15
## Change in soil water content (mm): 21.1829
## Soil water balance result (mm): 21.1829
## Change in snowpack water content (mm): 0
## Snowpack water balance result (mm): 0
## Water balance components:
##   Precipitation (mm) 1066 Rain (mm) 1066 Snow (mm) 0
##   Interception (mm) 141 Net rainfall (mm) 925
##   Infiltration (mm) 833 Infiltration excess (mm) 92 Saturation excess (mm) 272 Capillarity rise (mm)
##   Soil evaporation (mm) 21  Herbaceous transpiration (mm) 0 Woody plant transpiration (mm) 323
##   Plant extraction from soil (mm) 323  Plant water balance (mm) -0 Hydraulic redistribution (mm) 36
##   Runoff (mm) 364 Deep drainage (mm) 195
```

Step 15. Comparing the performance of the two models

To compare the performance of the two models with respect to observed data we can calculate the evaluation statistics for soil moisture:

```
evaluation_stats(fb_basic, fb_observed, type = "SWC")
```

```
##            n         Bias     Bias.rel          MAE      MAE.rel            r
## 364.00000000  -0.12212079 -28.47040214   0.12220446  28.48990865   0.91255597
##          NSE      NSE.abs
##  -0.01186163   0.04257990
```

```
evaluation_stats(fb_adv, fb_observed, type = "SWC")
```

```
##            n         Bias     Bias.rel          MAE      MAE.rel            r
## 364.00000000  -0.13293067 -30.99054249   0.13293067  30.99054249   0.94982546
##          NSE      NSE.abs
##  -0.09692368  -0.04145536
```

... and for stand evapotranspiration:

```
evaluation_stats(fb_basic, fb_observed, type = "SE+TR")
```

```
##            n         Bias     Bias.rel          MAE      MAE.rel            r
## 365.00000000  -0.40453685 -30.36201568   0.48384513  36.31440058   0.70498484
##          NSE      NSE.abs
##   0.03907060   0.02826233
```

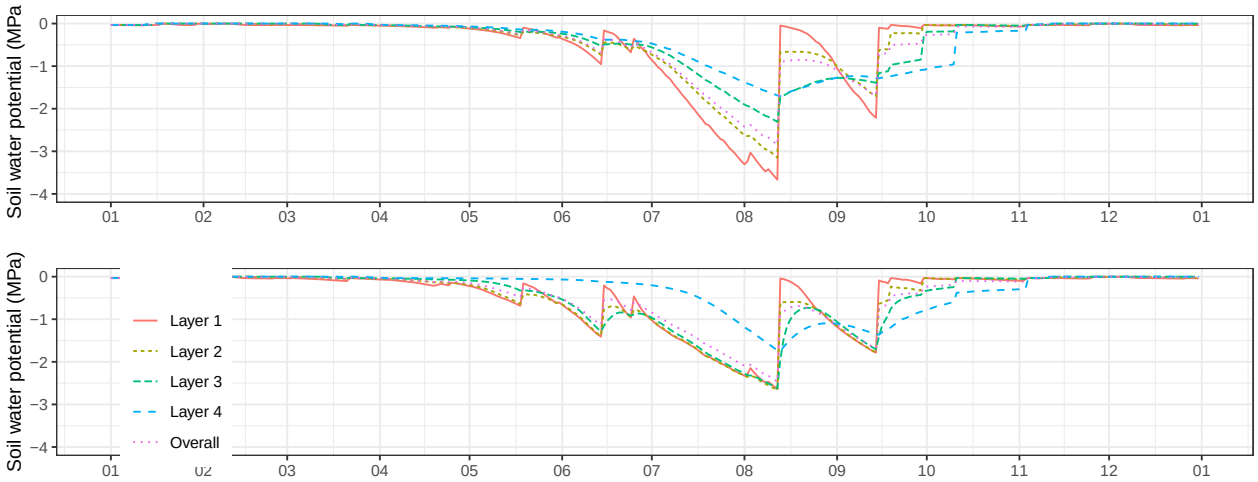
```
evaluation_stats(fb_adv, fb_observed, type = "SE+TR")
```

```
##            n          Bias      Bias.rel           MAE       MAE.rel
## 365.000000000  -0.388630403 -29.168176616   0.470984668  35.349174644
##             r           NSE       NSE.abs
##   0.691737785  -0.009257942   0.054090829
```

Step 16. Comparing soil moisture dynamics

We can compare soil layer moisture dynamics by drawing soil water potentials:

```
g1<-plot(fb_basic, "SoilPsi", ylim= c(-4,0))+
  scale_x_date(date_breaks = "1 month", date_labels = "%m")+theme(legend.position = "none")
g2<-plot(fb_adv, "SoilPsi", ylim= c(-4,0))+
  scale_x_date(date_breaks = "1 month", date_labels = "%m")+theme(legend.position = c(0.1,0.47))
plot_grid(g1,g2, ncol=1)
```

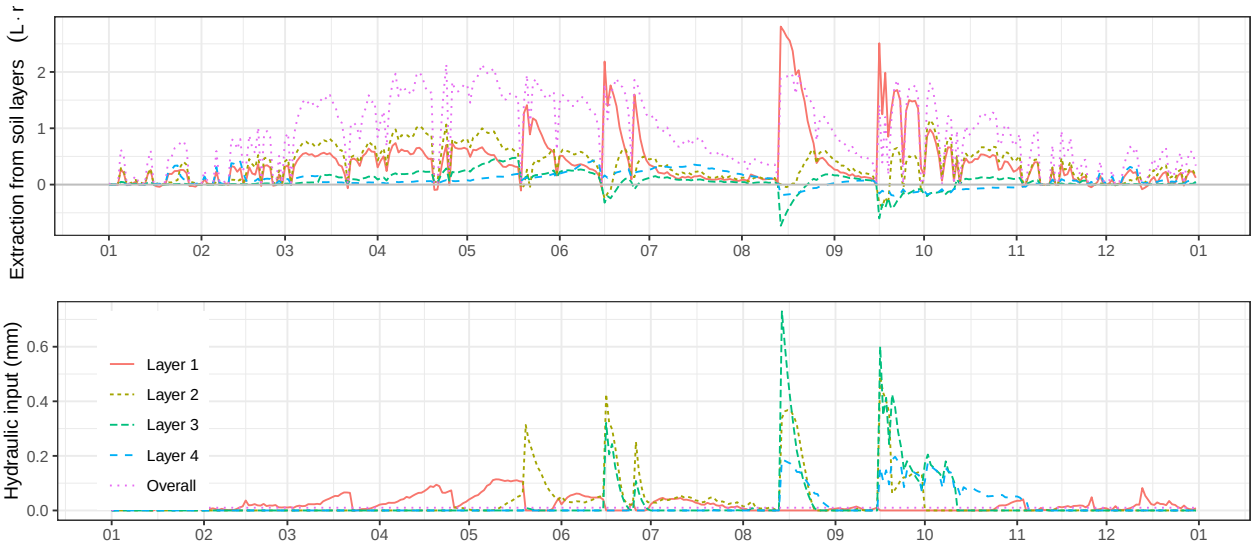


The basic model dries the soil more than the advanced model, which produces a stronger coupling between soil layers because of hydraulic redistribution.

Step 17. Understanding extraction and hydraulic redistribution

The following shows the daily root water uptake (or release) from different soil layers, and the daily amount of water entering soil layers due to hydraulic redistribution:

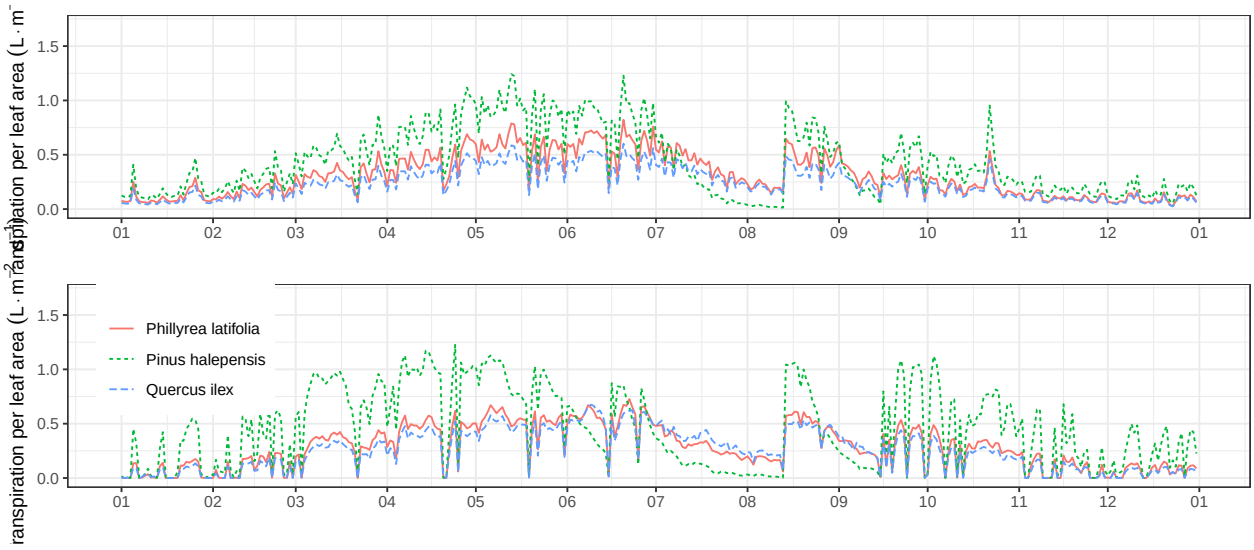
```
g1<-plot(fb_adv, "PlantExtraction")+
  scale_x_date(date_breaks = "1 month", date_labels = "%m")+theme(legend.position = "none")
g2<-plot(fb_adv, "HydraulicRedistribution")+
  scale_x_date(date_breaks = "1 month", date_labels = "%m")+theme(legend.position = c(0.08,0.5))
plot_grid(g1, g2, rel_heights = c(0.8,0.8), ncol=1)
```



Step 18. Comparing leaf-level transpiration dynamics

We can display the transpiration per leaf area unit basis using `"TranspirationPerLeaf"`.

```
g1<-plot(fb_basic, "TranspirationPerLeaf", bySpecies = TRUE, ylim = c(0,1.7))+
  scale_x_date(date_breaks = "1 month", date_labels = "%m")+theme(legend.position = "none")
g2<-plot(fb_adv, "TranspirationPerLeaf", bySpecies = TRUE, ylim = c(0,1.7))+
  scale_x_date(date_breaks = "1 month", date_labels = "%m")+theme(legend.position = c(0.1,0.7))
plot_grid(g1,g2, ncol=1)
```

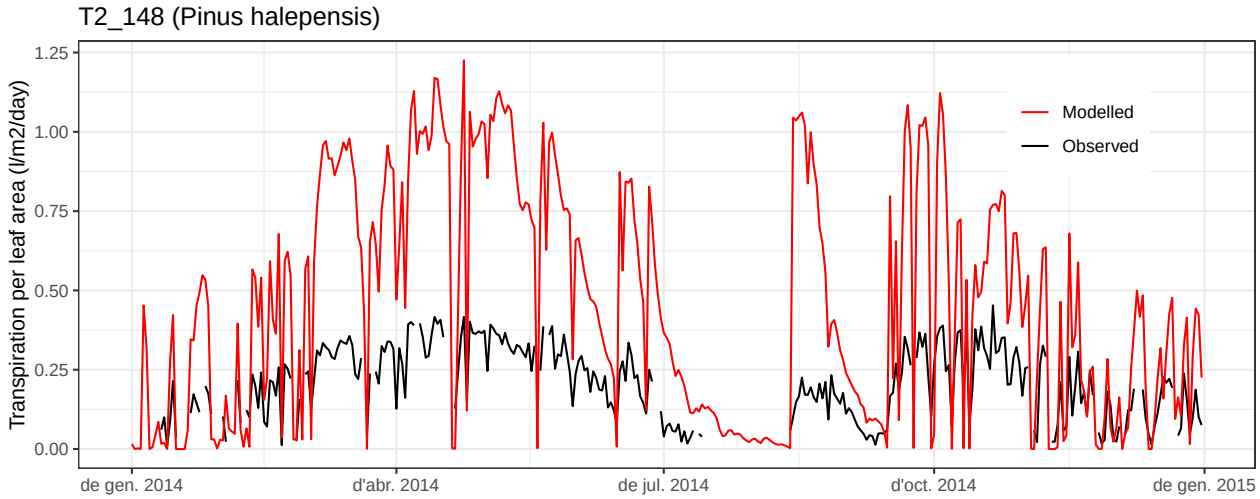


The basic model produces higher transpiration than the advanced model.

Step 19. Evaluation of tree transpiration

The following displays the observed and predicted transpiration for *Pinus halepensis* ...

```
evaluation_plot(fb_adv, fb_observed, cohort = "T2_148", type="E", plotType = "dynamics")+
  theme(legend.position = c(0.85,0.83))
```



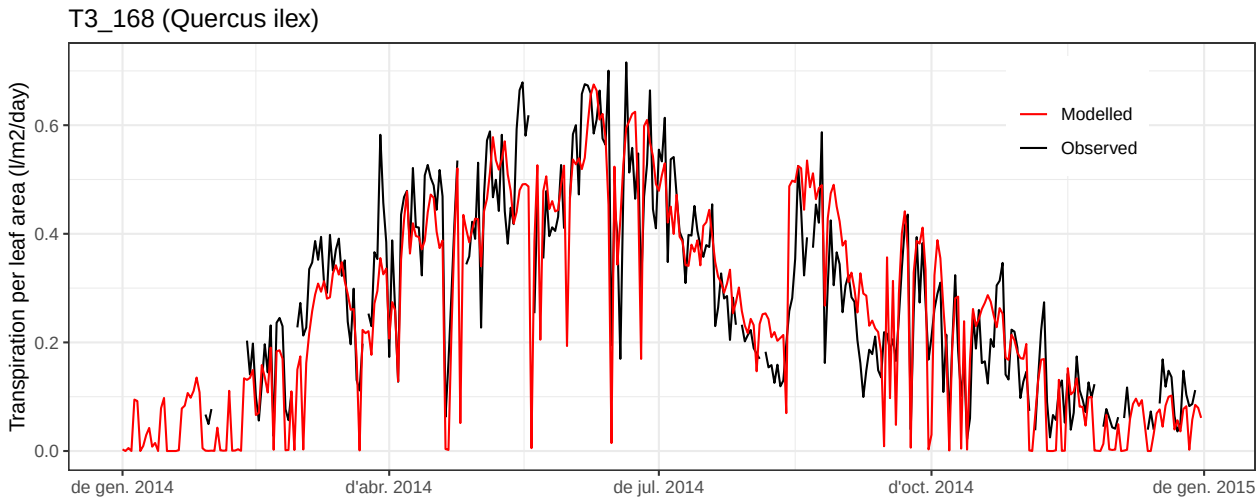

```
evaluation_stats(fb_adv, fb_observed, cohort = "T2_148", type="E")
```

```
##           n        Bias    Bias.rel         MAE     MAE.rel           r
## 300.0000000   0.3116129 151.5098933   0.3277188 159.3407793   0.8572620
##         NSE     NSE.abs
## -11.6137565  -2.3117672
```

Step 19. Evaluation of tree transpiration

... and *Quercus ilex*:

```
evaluation_plot(fb_adv, fb_observed, cohort = "T3_168", type="E", plotType = "dynamics")+
  theme(legend.position = c(0.85,0.83))
```



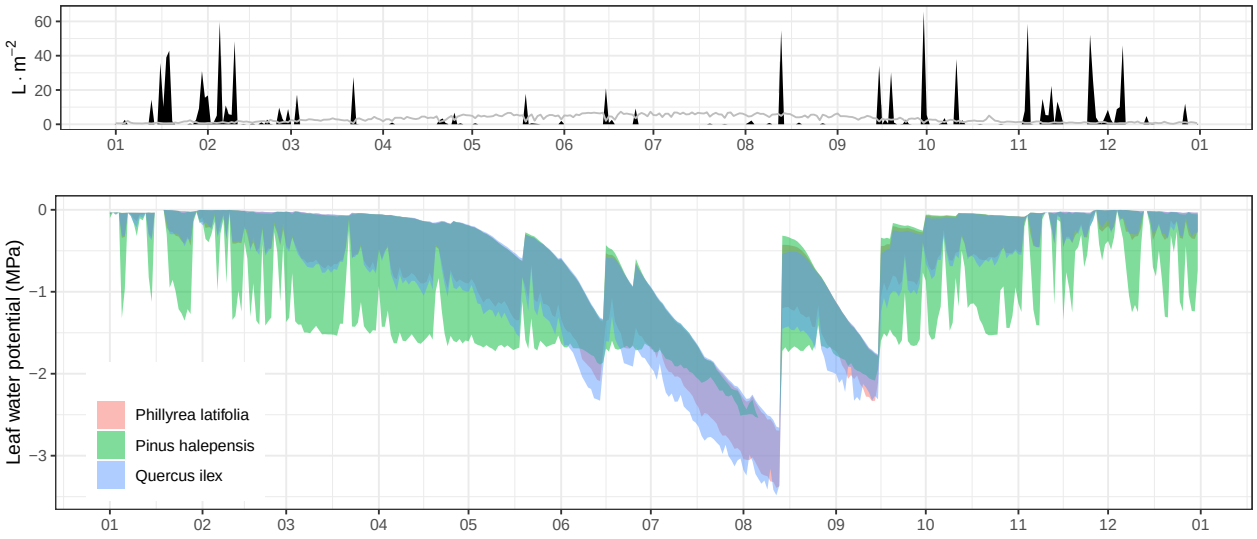
```
evaluation_stats(fb_adv, fb_observed, cohort = "T3_168", type="E")
```

```
##             n          Bias      Bias.rel           MAE       MAE.rel
## 309.000000000 -0.005350873 -1.848579701   0.066790546 23.074298510
##             r           NSE       NSE.abs
##   0.888794621   0.764822528   0.544970055
```

Step 20. Examining leaf water potentials

The following plots illustrate that the model simulates a tighter stomatal control for *Pinus halepensis*.

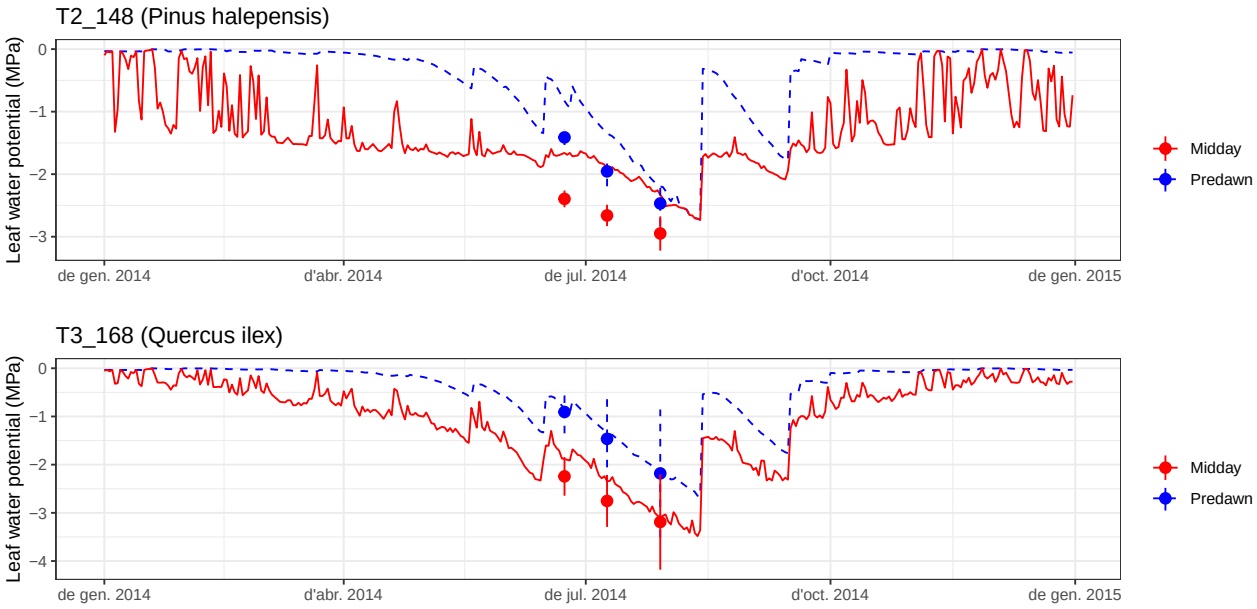
```
g1<-plot(fb_adv)+scale_x_date(date_breaks = "1 month", date_labels = "%m")+
  theme(legend.position = "none")
g2<-plot(fb_adv, "LeafPsiRange", bySpecies = TRUE)+
  scale_x_date(date_breaks = "1 month", date_labels = "%m")+
  theme(legend.position = c(0.1,0.25)) + ylab("Leaf water potential (MPa)")
plot_grid(g1, g2, ncol=1, rel_heights = c(0.4,0.8))
```



Step 21. Comparing leaf water potentials with observations

If we compare leaf water potentials against observations (`type = "WP"`) we obtain a rather good performance for *Q. ilex*, but midday water potentials are less well approximated for *P. halepensis*.

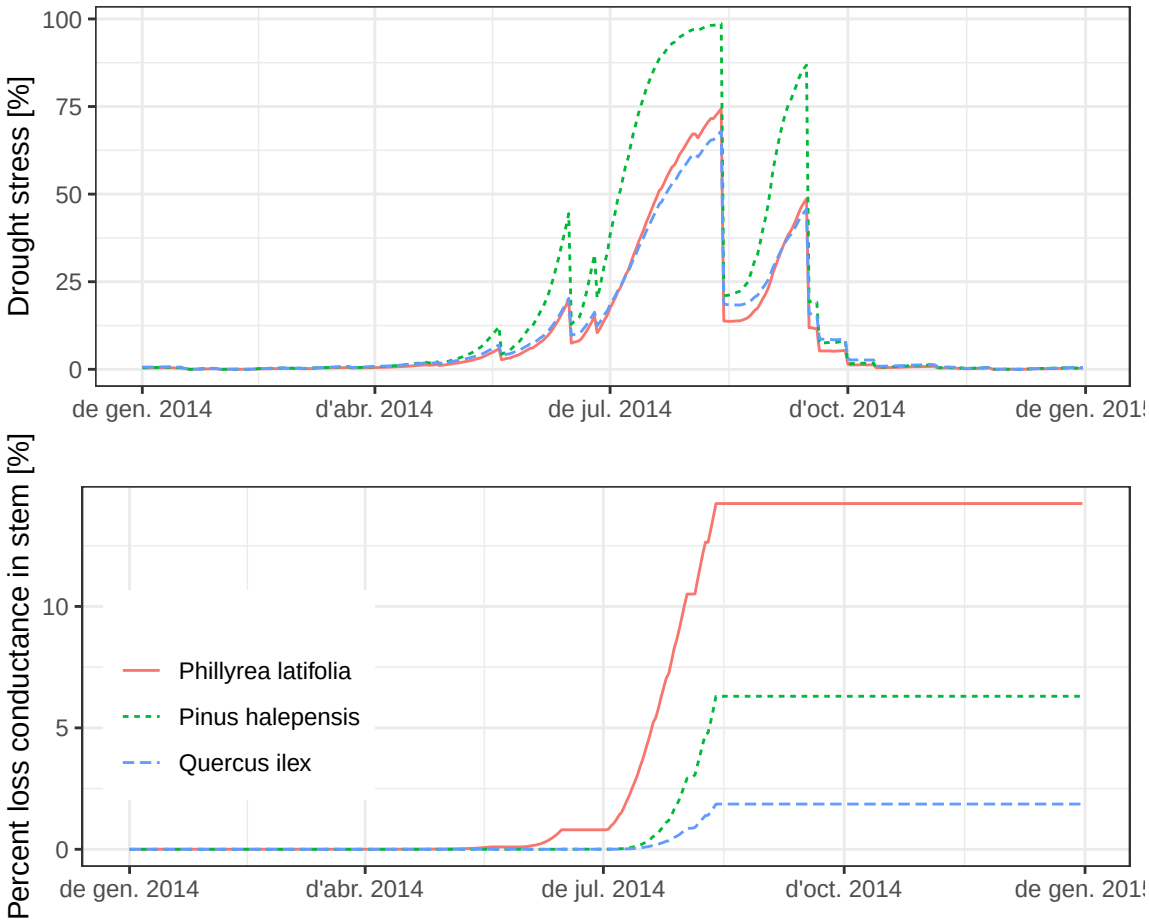
```
g1<-evaluation_plot(fb_adv, fb_observed, cohort = "T2_148", type="WP", plotType = "dynamics")
g2<-evaluation_plot(fb_adv, fb_observed, cohort = "T3_168", type="WP", plotType = "dynamics")
plot_grid(g1, g2, ncol=1)
```



Steps 22-23. Drought stress and PLC

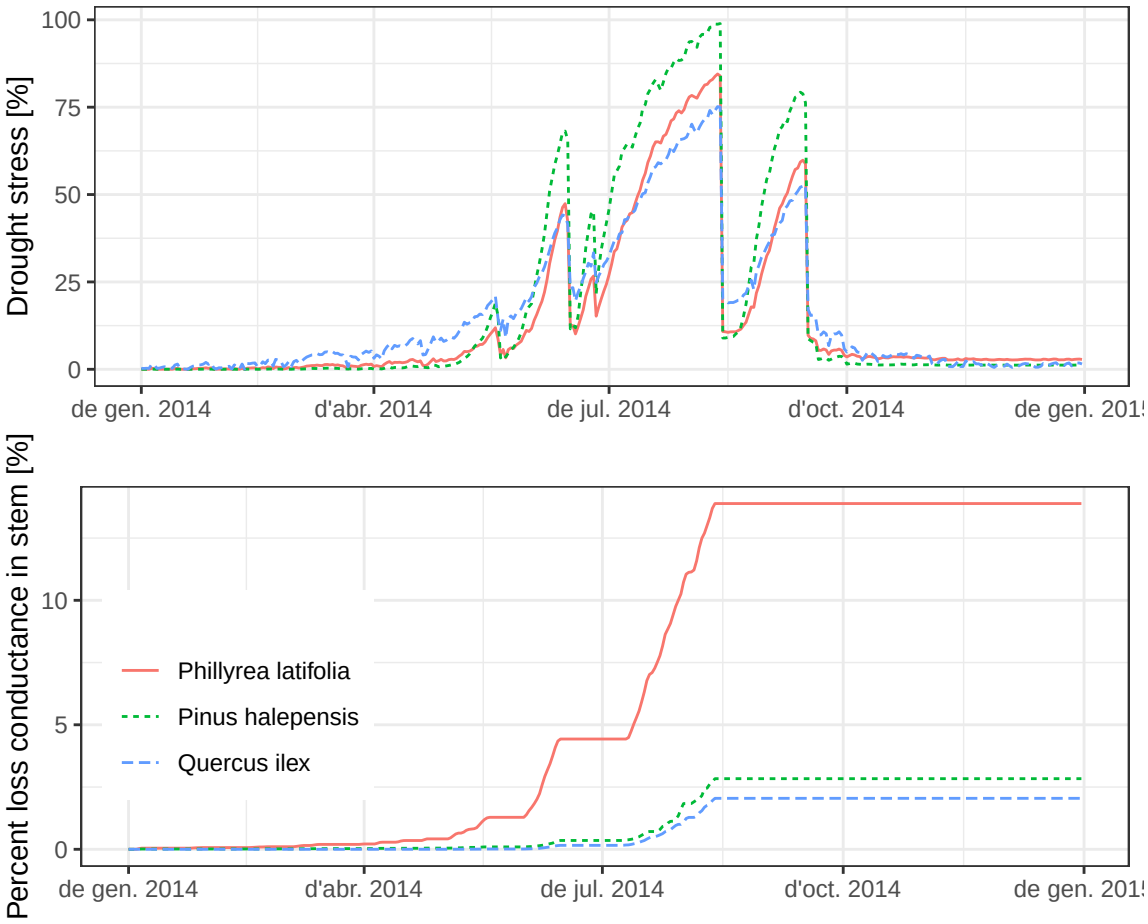
Basic model

```
g1<-plot(fb_basic, "PlantStress", bySpecies = TRUE)+
  theme(legend.position = "none")
g2<-plot(fb_basic, "StemPLC", bySpecies = TRUE)+
  theme(legend.position = c(0.15,0.45))
plot_grid(g1, g2, ncol=1)
```



Advanced model

```
g3<-plot(fb_adv, "PlantStress", bySpecies = TRUE)+
  theme(legend.position = "none")
g4<-plot(fb_adv, "StemPLC", bySpecies = TRUE)+
  theme(legend.position = c(0.15,0.45))
plot_grid(g3, g4, ncol=1)
```

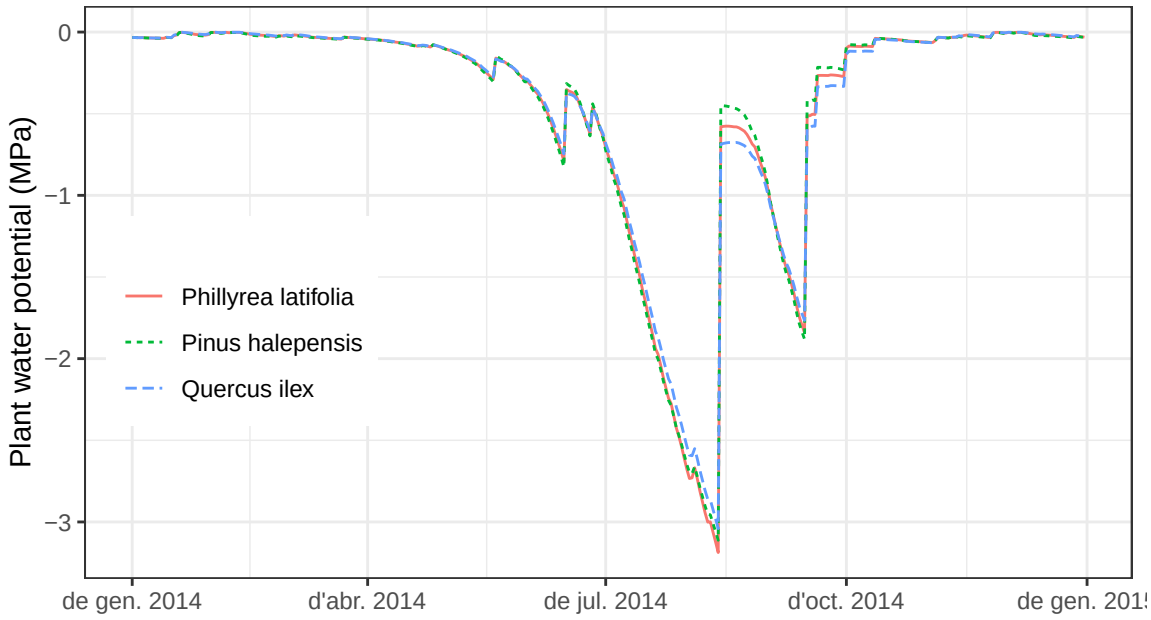


The basic model seems to overestimate PLC for *Pinus halepensis*, compared to the advanced model.

This could arise from a difference in the parameters determining PLC or differences in the water potential simulated by both models. We examine the first option using:

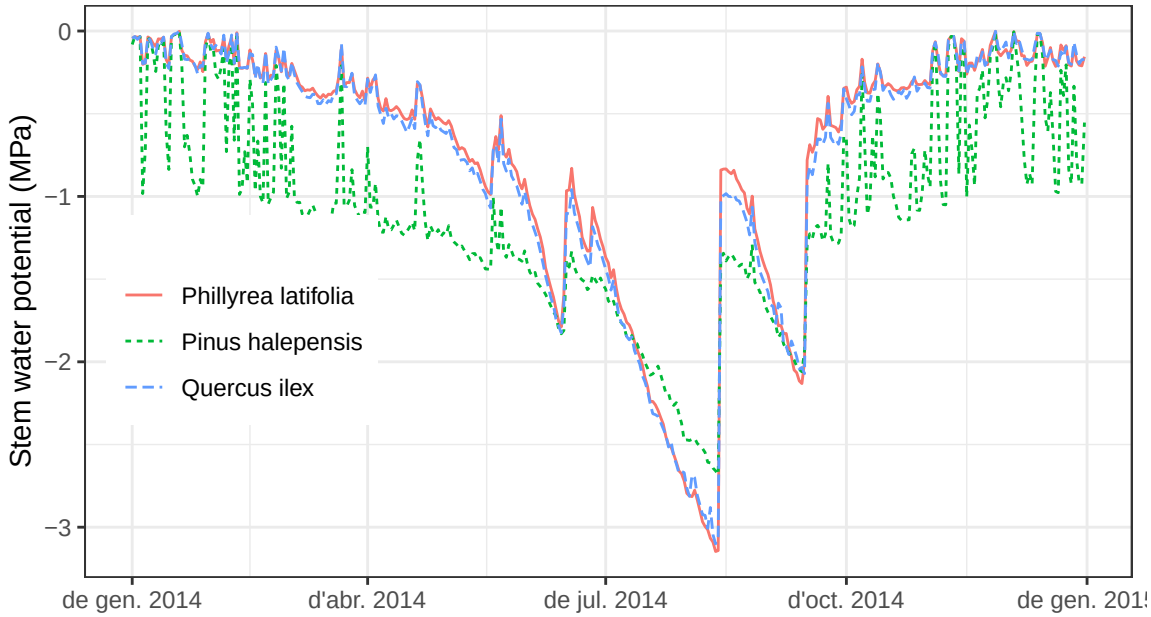
Basic model

```
plot(fb_basic, "PlantPsi", bySpecies = TRUE)+
  theme(legend.position = c(0.15,0.45))
```



Advanced model

```
plot(fb_adv, "StemPsi", bySpecies = TRUE)+
  theme(legend.position = c(0.15,0.45))
```



The two models produce similar overall patterns, but the advanced model predicts less negative water potentials for *P. halepensis* in summer.