Fitting Process Models
Part 2: Fitting a (simple) process model: CO_2 and H_2O fluxes over boreal forests

Carsten F. Dormann[1]

[1]Biometry & Environmental System Analysis, University of Freiburg, Germany
carsten.dormann@biom.uni-freiburg.de

July 13, 2019

There are several issues to be concerned with when fitting a process model. As illustration it may suffice to use an actual (rather than a toy) model with a very limited number of parameters: PRELES. We fit it to daily measurements of gas exchange in a site in Finland (the home of PRELES).

Contents

1 *Introduction*

PRELES (PREdicting Light-use efficiency, Evapotranspiration and Soil water content) is a three-compartment "ecosystem model" (Peltoniemi et al., 2015; Minunno et al., 2016). This is not the place to present the model's equations, but it is entirely deterministic and consists of about 15 equations to describe light capture, water loss and the soil water component (Fig. 1).

PRELES is written in C++ and conveniently available through the **Rpreles**-package. Parameter ranges and data for four Finnish field sites are available in the .Rdata-files accompanying this file.

A sensitivity analysis should preceed the fitting, but is omitted here for sake of focus on the fitting itself.

Let's start by installing the relevant package, data and a file with default parameter settings.

```
devtools::install_github("MikkoPeltoniemi/Rpreles")  # downloads and compiles PRELES
```

```
library(Rpreles)
library(BayesianTools)
load("EddyCovarianceDataBorealSites.rdata")  # s1-4
load("parameterRanges.rdata")  # par
```

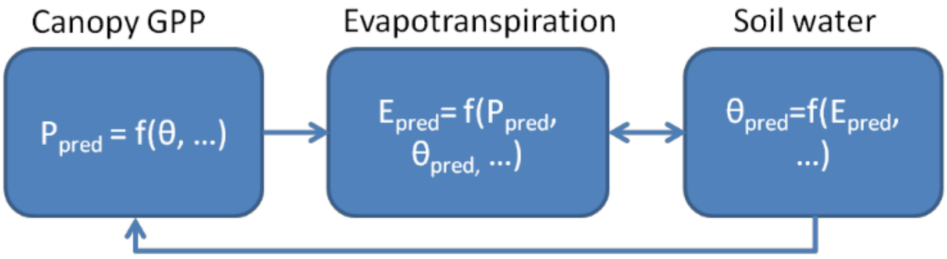


Figure 1: Extremely simplified composition of PRELES (nicked from a MSc-student's report: thanks L.G.!).

```
`?`(PRELES)
```

The environmental data look like this:

```
head(s1)
```

```
        PAR   TAir   VPD Precip CO2     fAPAR GPPobs  ETobs DOY
1 0.3951072 -9.328 0.010    0.7 380 0.8039803  0.000 -0.056   1
2 0.1573344 -3.484 0.016    1.6 380 0.8039803  0.100  0.061   2
3 0.2306016 -1.894 0.011    7.0 380 0.8039803  0.013  0.050   3
4 0.4738176  0.373 0.017    0.8 380 0.8039803  0.033 -0.002   4
5 0.9157536 -1.085 0.031    2.3 380 0.8039803  0.071 -0.013   5
6 0.4288896 -0.089 0.036    0.1 380 0.8039803  0.079 -0.037   6
```

```
attach(s1)
```

```
The following object is masked from package:datasets:

    CO2
```

Let's look at the PRELES-parameters:

```
par  # note that '-999' is supposed to indiate NA!
```

```
           name         def       min       max
1     soildepth  413.000000 -9.99e+02 -999.0000
2       ThetaFC    0.450000 -9.99e+02 -999.0000
3      ThetaPWP    0.118000 -9.99e+02 -999.0000
4   tauDrainage    3.000000 -9.99e+02 -999.0000
5          beta    0.748018  2.00e-01    2.5000
6           tau   13.233830  1.00e+00   25.0000
7          X[0]   -3.965787 -2.00e+01   20.1000
8        S[max]   18.766960  2.30e+00   30.0000
9         kappa   -0.130473 -1.00e+00   -0.0010
10        gamma    0.034459  1.03e-04    0.5030
11       rho[P]    0.450828  0.00e+00    0.9990
12        cmCO2 2000.000000 -9.99e+02 -999.0000
13    ckappaCO2    0.400000 -9.99e+02 -999.0000
14        alpha    0.324463  1.00e-06   10.0000
15       lambda    0.874151  1.00e-04    1.2000
16          chi    0.075601  0.00e+00    2.5000
17      rho[ET]    0.541605  0.00e+00    0.9999
18           nu    0.273584  1.00e-04    5.0000
19     Meltcoef    1.200000 -9.99e+02 -999.0000
20          I_0    0.330000 -9.99e+02 -999.0000
21        CWmax    4.970496 -9.99e+02 -999.0000
22 SnowThreshold    0.000000 -9.99e+02 -999.0000
23          T_0    0.000000 -9.99e+02 -999.0000
24       SWinit  200.000000 -9.99e+02 -999.0000
25       CWinit    0.000000 -9.99e+02 -999.0000
26      SOGinit    0.000000 -9.99e+02 -999.0000
27        Sinit   20.000000 -9.99e+02 -999.0000
28           t0 -999.000000 -9.99e+02 -999.0000
```

```
29         tcrit -999.000000 -9.99e+02 -999.0000
30      tsumcrit -999.000000 -9.99e+02 -999.0000
31        sd_gpp    1.000000  1.00e-03    5.0000
32         sd_et    1.000000  0.00e+00   10.0000
```

```
par[par == "-999"] <- NA
par
```

```
           name         def       min     max
1     soildepth  413.000000        NA      NA
2       ThetaFC    0.450000        NA      NA
3      ThetaPWP    0.118000        NA      NA
4   tauDrainage    3.000000        NA      NA
5          beta    0.748018  2.00e-01  2.5000
6           tau   13.233830  1.00e+00 25.0000
7          X[0]   -3.965787 -2.00e+01 20.1000
8        S[max]   18.766960  2.30e+00 30.0000
9         kappa   -0.130473 -1.00e+00 -0.0010
10        gamma    0.034459  1.03e-04  0.5030
11       rho[P]    0.450828  0.00e+00  0.9990
12        cmCO2 2000.000000        NA      NA
13    ckappaCO2    0.400000        NA      NA
14        alpha    0.324463  1.00e-06 10.0000
15       lambda    0.874151  1.00e-04  1.2000
16          chi    0.075601  0.00e+00  2.5000
17      rho[ET]    0.541605  0.00e+00  0.9999
18           nu    0.273584  1.00e-04  5.0000
19     Meltcoef    1.200000        NA      NA
20          I_0    0.330000        NA      NA
21        CWmax    4.970496        NA      NA
22 SnowThreshold    0.000000        NA      NA
23          T_0    0.000000        NA      NA
24       SWinit  200.000000        NA      NA
25       CWinit    0.000000        NA      NA
26      SOGinit    0.000000        NA      NA
27        Sinit   20.000000        NA      NA
28           t0          NA        NA      NA
29        tcrit          NA        NA      NA
30     tsumcrit          NA        NA      NA
31       sd_gpp    1.000000  1.00e-03  5.0000
32        sd_et    1.000000  0.00e+00 10.0000
```

In this case, several parameters are actually site-specific constants, such as soil depth. Also, the last two parameters were added for our calibration; they are not required for PRELES, but serve as placeholder for the standard deviation of a normal distribution used with **BayesianTools** later.

2 *Running PRELES with default parameters*

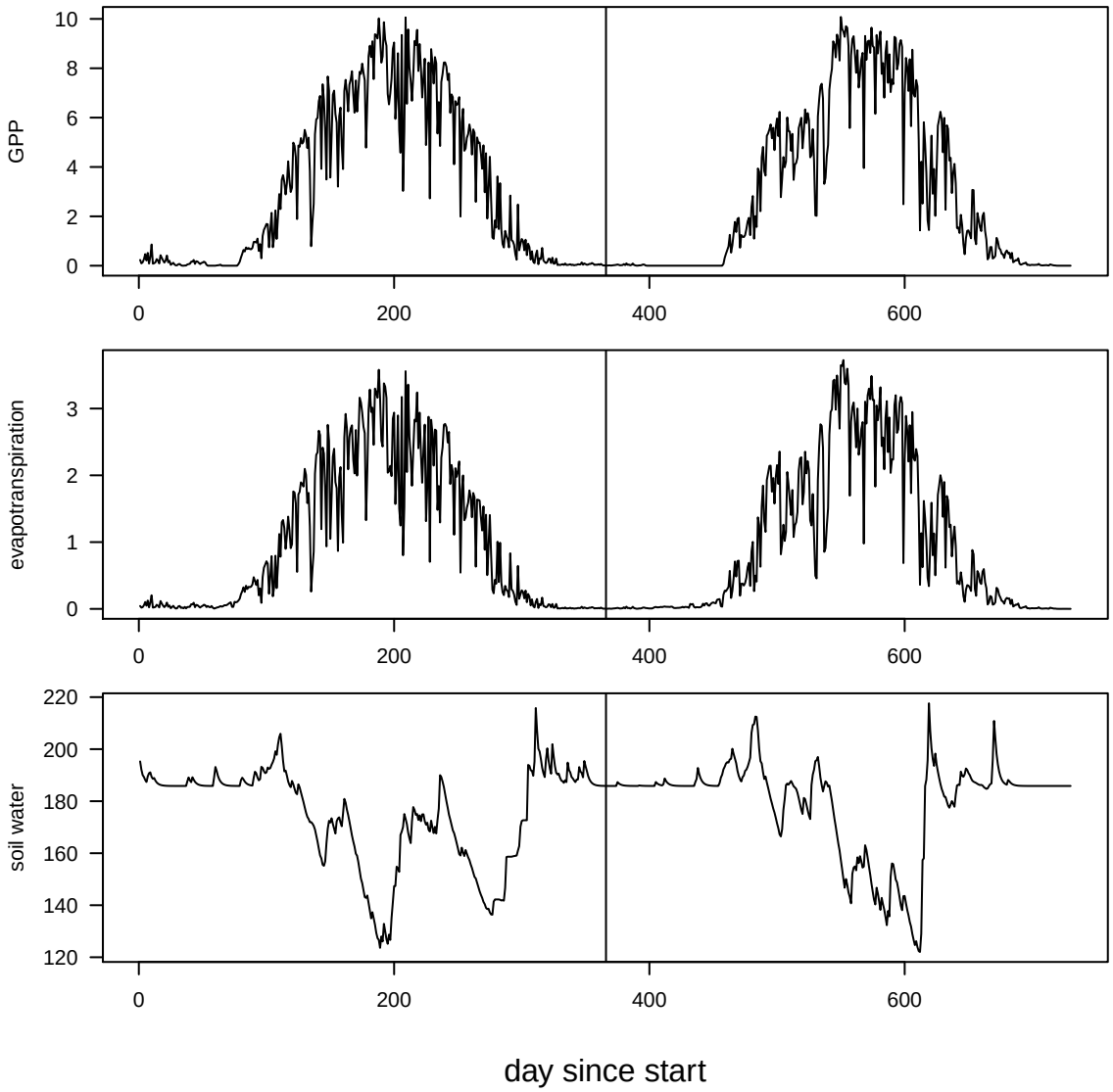
PRELES requires a number of arguments (essentially the drivers, the parameter vector p and various settings). The minimal run, using the environmental data from site Hyytiälä (`s1`), may look like this:

```
onerun <- PRELES(PAR = PAR, TAir = TAir, VPD = VPD, Precip = Precip, CO2 = CO2,
    fAPAR = fAPAR, p = par[, "def"])
str(onerun)
```

```
List of 3
 $ GPP: num [1:730] 0.2341 0.0939 0.1374 0.2636 0.4694 ...
 $ ET : num [1:730] 0.0441 0.0189 0.0266 0.0541 0.1036 ...
 $ SW : num [1:730] 195 192 190 189 188 ...
```

```
# make a plot of the output:
par(mfrow = c(3, 1), mar = c(2, 4, 1, 1), oma = c(4, 0, 0, 0))
plot(1:(2 * 365), onerun$GPP, type = "l", las = 1, ylab = "GPP")
abline(v = 366)
```

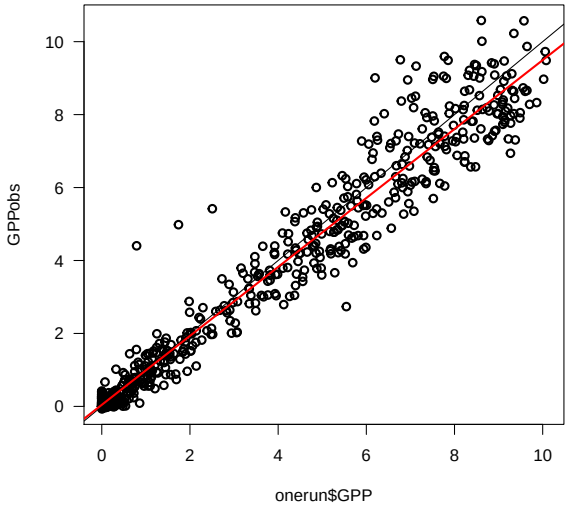
```
plot(1:(2 * 365), onerun$ET, type = "l", las = 1, ylab = "evapotranspiration")
abline(v = 366)
plot(1:(2 * 365), onerun$SW, type = "l", las = 1, ylab = "soil water")
abline(v = 366)
mtext(side = 1, line = 4, "day since start")
```



A comparison of, say, GPP with observed values shows relatively little room for improvement:

```
plot(onerun$GPP, GPPobs, las = 1, lwd = 2)
abline(0, 1)
abline(lm(GPPobs ~ onerun$GPP), col = "red", lwd = 2)
RMSE ← function(x, y) sqrt(mean((x − y)^2))
RMSE(onerun$GPP, GPPobs)
```

```
[1] 0.6688096
```



3 *Bayesian calibration to site 1*

To fit PRELES to the observed data, we employ the **BayesianTools** framework. This requires us to set up the likelihood and priors first. We shall fit the model to the GPP data (to start with).

First, we duplicate the parameters, to have one working version, to be modified, and our original parameters untouched.

We select a few parameters to be fitted, and leave the constants in place. To do so, I create a vector of "parameters to change".

```
library(BayesianTools)
# select the parameters to be calibrated:
thispar ← par$def
names(thispar) ← par$name

pars2tune ← c(5:11, 14:18, 31)  # note that we omit 32, as it refers to ET
```

Then we have to define the likelihood for our friend **BayesianTools**. As a first pass, we want to fit the model to the observed gross primary productivity (GPP), which, as you have seen above, is one of the outputs of PRELES.

And we have to do the other stuff, i.e. set up the priors and the MCMC-settings. If you are unclear about this, check last session's example or the package's help. In this case, we use uniform priors, and use the conveniently provided upper and lower bounds of the parameter table.[1]

```
ell ← function(pars) {
    # pars is a vector the same length as pars2tune
    thispar[pars2tune] ← pars
    # likelihood function, first shot: normal density
    with(s1, sum(dnorm(GPPobs, mean = PRELES(PAR = PAR, TAir = TAir, VPD = VPD,
        Precip = Precip, CO2 = CO2, fAPAR = fAPAR, p = thispar)$GPP, sd = thispar[31],
        log = T)))
}
priors ← createUniformPrior(lower = par$min[pars2tune], upper = par$max[pars2tune],
    best = par$def[pars2tune])
setup ← createBayesianSetup(likelihood = ell, prior = priors, parallel = T)
```

```
parallel function execution created with3cores.
```

```
settings ← list(iterations = 1e+05, parallel = T, message = F)  # for DEzs, suppress output
```

[1] Long and interesting stories can be told about priors. Not by me, though. Have a look, if you want, at (Lemoine, 2019), for some recent thoughts on moderately informative priors.

(Switching off the reporting is done here to avoid filling many pages with rather boring information. Normally, I would always set to to T, the default, to see that it is still running.)

Now we run the actual sampler:

```
set.seed(1)
fit1 <- runMCMC(bayesianSetup = setup, settings = settings, sampler = "DEzs")
```

We have here used the so-called "differential evolution MCMC with snooker updates", which is cool, and fast. Like DE itself, its chains are informing each other, thereby more efficiently searching parameter space than our old friend Metropolis.

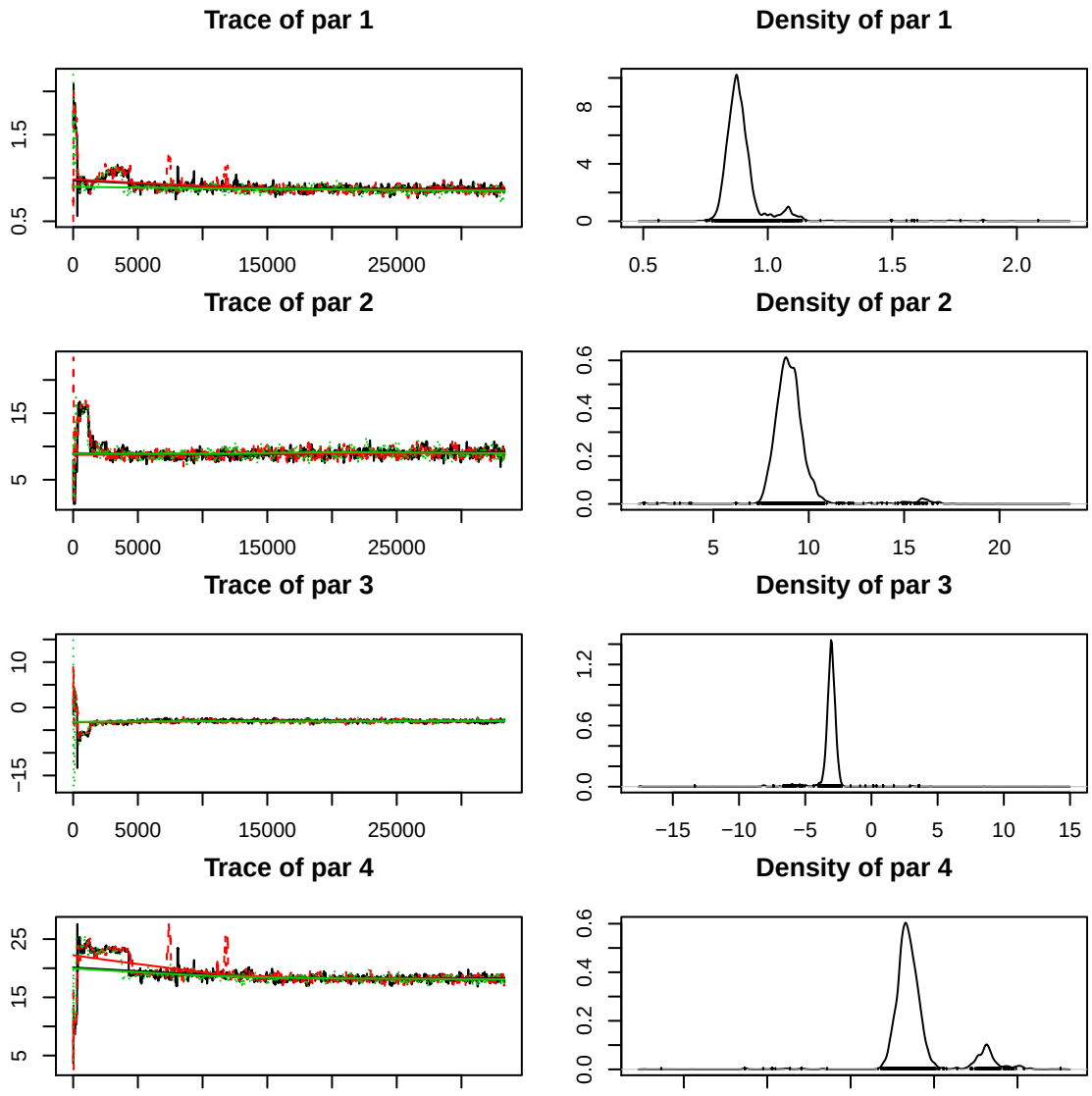
Whatever. Let's look at the results.

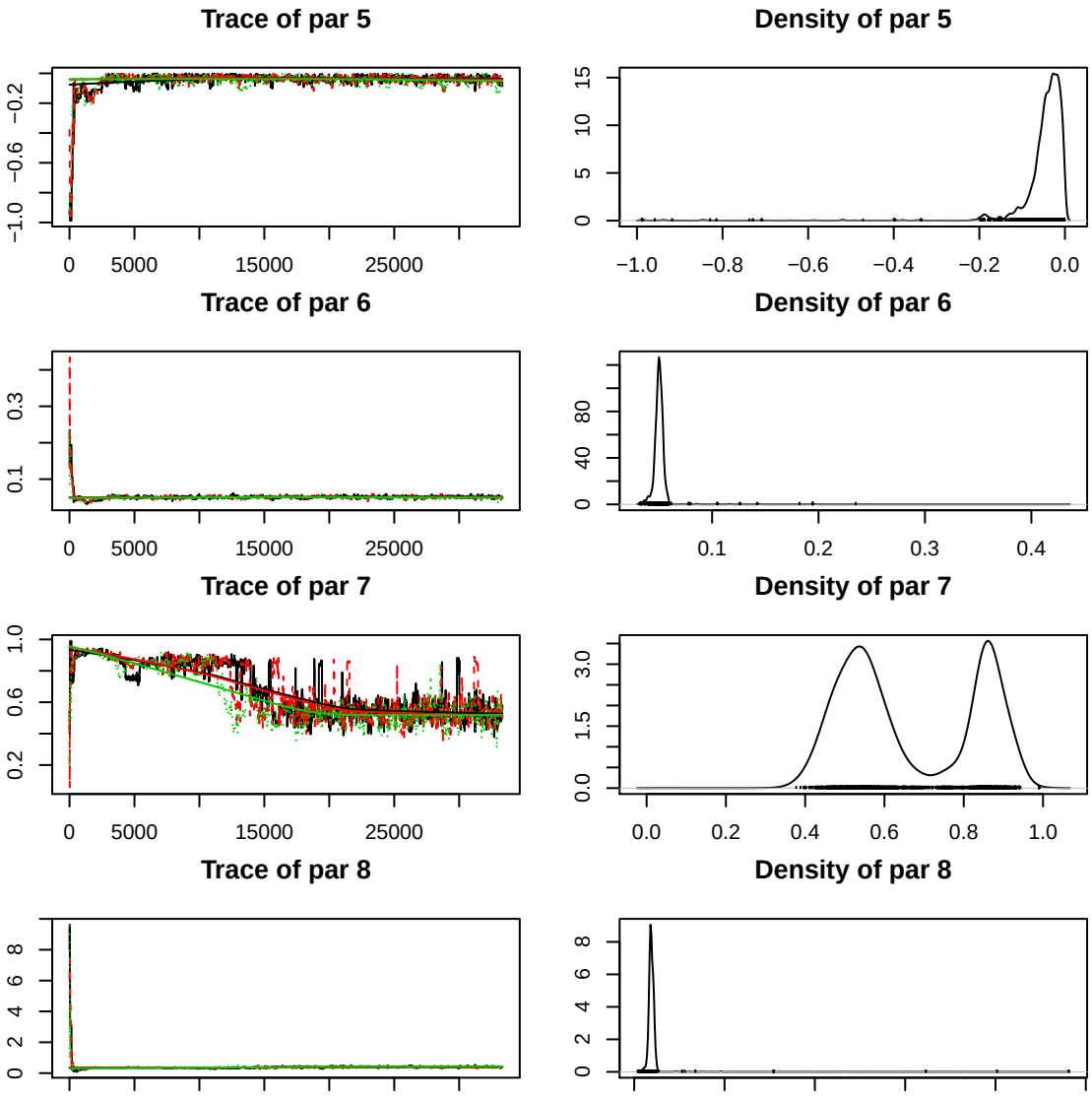
```
summary(fit1)
```

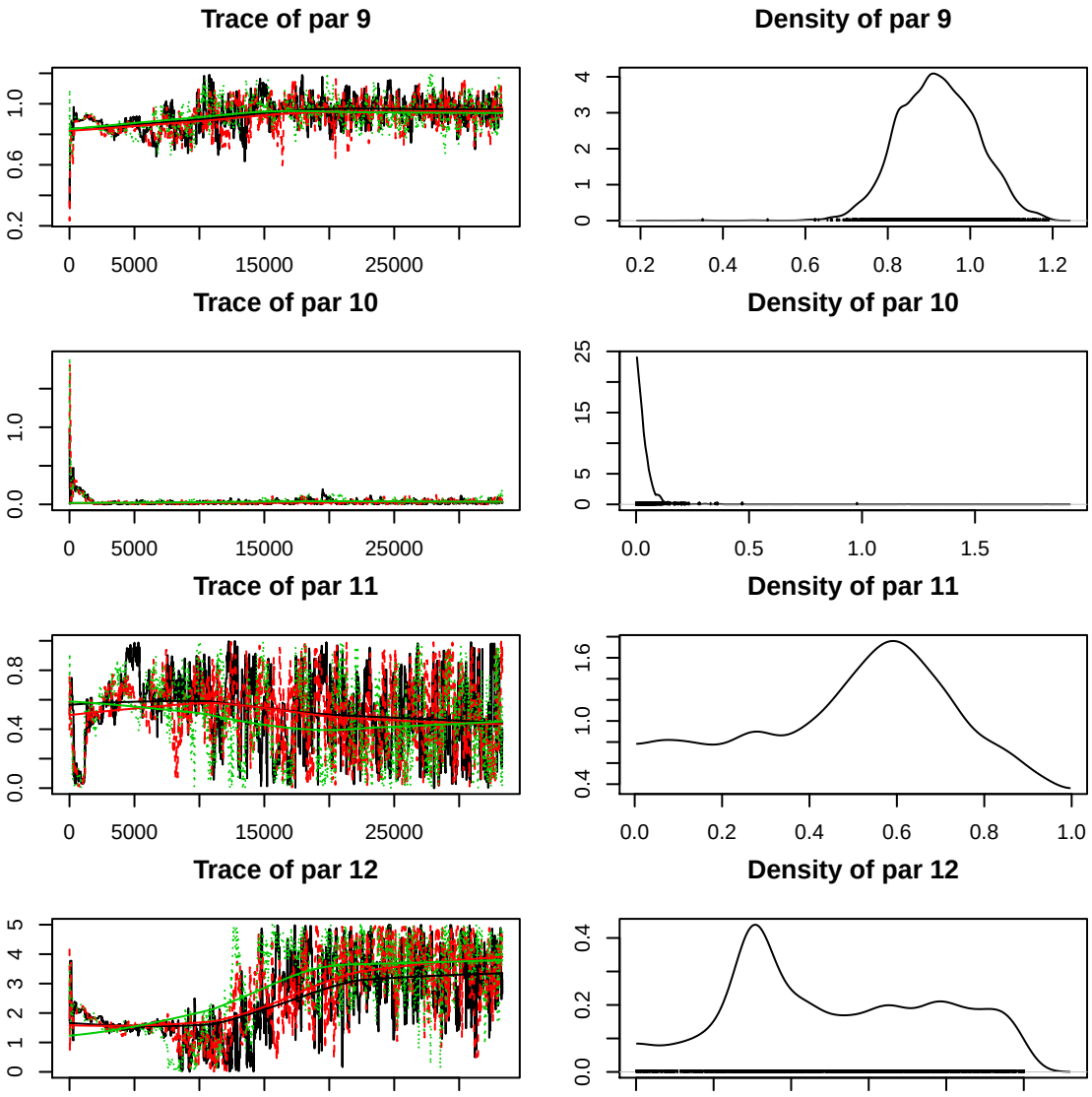
```
# # # # # # # # # # # # # # # # # # # # # # # # # # # #
## MCMC chain summary ##
# # # # # # # # # # # # # # # # # # # # # # # # # # # #

# MCMC sampler:  DEzs
# Nr. Chains:  3
# Iterations per chain:  33334
# Rejection rate:  0.936
# Effective sample size:  327
# Runtime:  114.108  sec.

# Parameters
         psf    MAP   2.5% median  97.5%
par 1  1.022  0.898  0.807  0.880  1.104
par 2  1.011  9.195  7.800  8.960 13.803
par 3  1.013 -2.974 -5.235 -3.039 -2.448
par 4  1.003 18.365 17.273 18.501 23.660
par 5  1.039 -0.018 -0.174 -0.036 -0.003
par 6  1.037  0.054  0.040  0.050  0.058
par 7  1.040  0.556  0.426  0.596  0.926
par 8  1.025  0.401  0.270  0.376  0.481
par 9  1.019  0.999  0.742  0.920  1.103
par 10 1.045  0.012  0.001  0.024  0.152
par 11 1.008  0.757  0.032  0.532  0.933
par 12 1.031  3.178  0.314  2.353  4.870
par 13 1.003  0.590  0.568  0.596  0.675

## DIC:  1358.413
## Convergence
 Gelman Rubin multivariate psrf:

## Correlations
        par 1  par 2  par 3  par 4  par 5  par 6  par 7  par 8  par 9
par 1   1.000 -0.125  0.286  0.130 -0.600  0.618  0.392  0.423 -0.338
par 2  -0.125  1.000 -0.492  0.458  0.038 -0.236  0.091 -0.146  0.042
par 3   0.286 -0.492  1.000 -0.558 -0.127  0.375 -0.111  0.338 -0.089
par 4   0.130  0.458 -0.558  1.000  0.216 -0.419  0.569 -0.374 -0.190
par 5  -0.600  0.038 -0.127  0.216  1.000 -0.622 -0.103 -0.618  0.146
par 6   0.618 -0.236  0.375 -0.419 -0.622  1.000 -0.066  0.795 -0.149
par 7   0.392  0.091 -0.111  0.569 -0.103 -0.066  1.000 -0.144 -0.326
par 8   0.423 -0.146  0.338 -0.374 -0.618  0.795 -0.144  1.000 -0.204
par 9  -0.338  0.042 -0.089 -0.190  0.146 -0.149 -0.326 -0.204  1.000
par 10  0.319  0.288 -0.075 -0.104 -0.561  0.517 -0.060  0.537 -0.083
par 11  0.113 -0.273  0.162 -0.015  0.043  0.074  0.172  0.013 -0.043
par 12 -0.267  0.061  0.002 -0.340  0.009  0.033 -0.815  0.116  0.237
par 13  0.676 -0.066  0.149 -0.318 -0.843  0.731  0.069  0.647 -0.188
       par 10 par 11 par 12 par 13
par 1   0.319  0.113 -0.267  0.676
par 2   0.288 -0.273  0.061 -0.066
par 3  -0.075  0.162  0.002  0.149
par 4  -0.104 -0.015 -0.340 -0.318
par 5  -0.561  0.043  0.009 -0.843
```

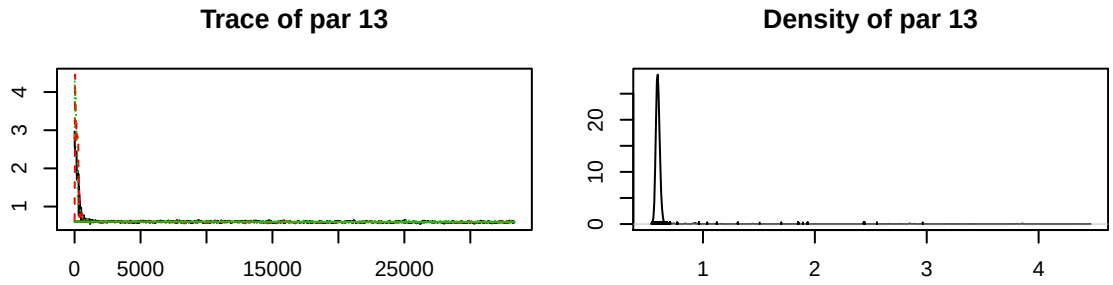
```
par 6   0.517  0.074  0.033  0.731
par 7  -0.060  0.172 -0.815  0.069
par 8   0.537  0.013  0.116  0.647
par 9  -0.083 -0.043  0.237 -0.188
par 10  1.000 -0.097  0.107  0.621
par 11 -0.097  1.000 -0.283 -0.016
par 12  0.107 -0.283  1.000 -0.034
par 13  0.621 -0.016 -0.034  1.000
```

```
par(mar = c(1, 4, 4, 0))
plot(fit1)
```




Trace of par 5
Density of par 5
Trace of par 6
Density of par 6
Trace of par 7
Density of par 7
Trace of par 8
Density of par 8

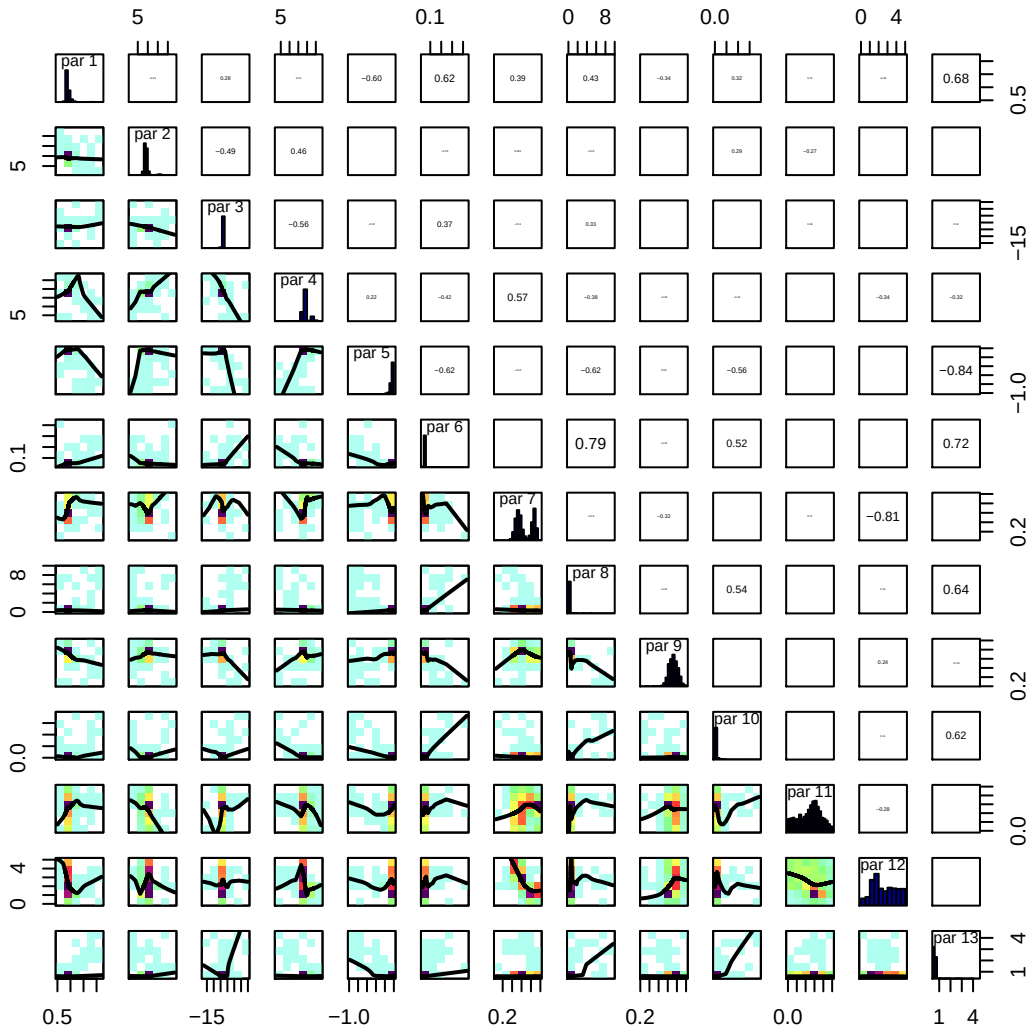

Trace of par 9
Density of par 9
Trace of par 10
Density of par 10
Trace of par 11
Density of par 11
Trace of par 12
Density of par 12



Oh, dear, that doesn't look very promising! The individual chains seem to be far from converging, but, hey, the posterior is a not too bad. And the Rhat-values are not too far from the desired 1 (or acceptable 1.1).

The correlation plot is rather difficult to read, with so many parameters, but it suggests no substantial correlation among them (see also the summary output).

```
correlationPlot(fit1)
```



The obvious thing to do, before the next step, would be to run the algorithm *much* longer.[2]

4 *Investigation of residuals*

The result of any MCMC-fit is a long list of parameter combinations. We use all (or at least a large number) of them for prediction and then average these predictions to compare with the observed values.[3]

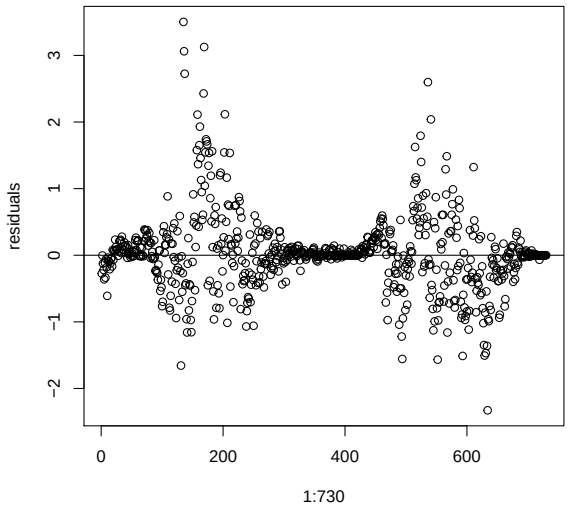
```
library(BayesianTools)  # don't ask why I have to explicitly load it again ...
library(Rpreles)
parSamples ← getSample(fit1, thin = 10)
# now run prediction with each parameter combination
getPreds ← function(pars) {
    thispar ← par$def
    thispar[pars2tune] ← pars
    preds ← with(s1, PRELES(PAR = PAR, TAir = TAir, VPD = VPD, Precip = Precip,
        CO2 = CO2, fAPAR = fAPAR, p = thispar)$GPP)
    return(preds)
}
preds ← sapply(1:nrow(parSamples), function(x) getPreds(parSamples[x,
    ]))
RMSE(rowMeans(preds), GPPobs)
```

[2] But we don't do that right now. Something for later, at home, over a glass of non-alcoholic beverage.

[3] Using the MAP is an option, too, but resorting to a point estimate does not chime well with a Bayesian approach.

```
[1] 0.588639
```

```
residuals ← GPPobs − rowMeans(preds)
plot(1:730, residuals)
abline(h = 0)
```



First: remember the RMSE of the default parameterisation: 0.67. This one is better!

But clearly variance is not constant in time. Rather it seems to be much higher in summer, when also the absolute fluxes are higher. This requires going back to the likelihood function and adapting the standard deviation there, e.g. by making it a function of GPP itself!

5 *Adapting the likelihood*

The next step may seem a bit, well, arbitrary. We will make the variance a function of the model's GPP output. It is not bad practice or fudging. But as statistically minded people we may never have encountered modelling of the variance before, and hence may find it bizarre. Well, it isn't. In GLMMs, one can also specify how the variance changes with the mean (the `varIdent`-argument in `nlme::lme`). And the purpose is mainly to find a function to optimise that will lead us to the best possible parameters.

Let's try; instead of making sd a constant, we make it a linear function of predicted GPP. We now usurp parameter 32 as slope of that relationship, and our original parameter 31 is the intercept.

```
pars2tune2 ← c(pars2tune, 32)
ellhetero ← function(pars) {
    # pars is a vector the same length as pars2tune
    thispar[pars2tune2] ← pars
    run ← with(s1, PRELES(PAR = PAR, TAir = TAir, VPD = VPD, Precip = Precip,
        CO2 = CO2, fAPAR = fAPAR, p = thispar))
    # likelihood function, second shot: normal density with heterogeneous
    # error
    sum(dnorm(s1$GPPobs, mean = run$GPP, sd = thispar[31] + thispar[32] *
        run$GPP, log = T))
}
# adapt the priors (include parameter 32)
priors2 ← createUniformPrior(lower = par$min[pars2tune2], upper = par$max[pars2tune2],
    best = par$def[pars2tune2])
# adapt the setup
setup2 ← createBayesianSetup(likelihood = ellhetero, prior = priors2,
    parallel = T)
```

```
parallel function execution created with3cores.
```

```r
# rerun the model
set.seed(2)
fit2 <- runMCMC(bayesianSetup = setup2, settings = settings, sampler = "DEzs")
```

```r
summary(fit2)
```

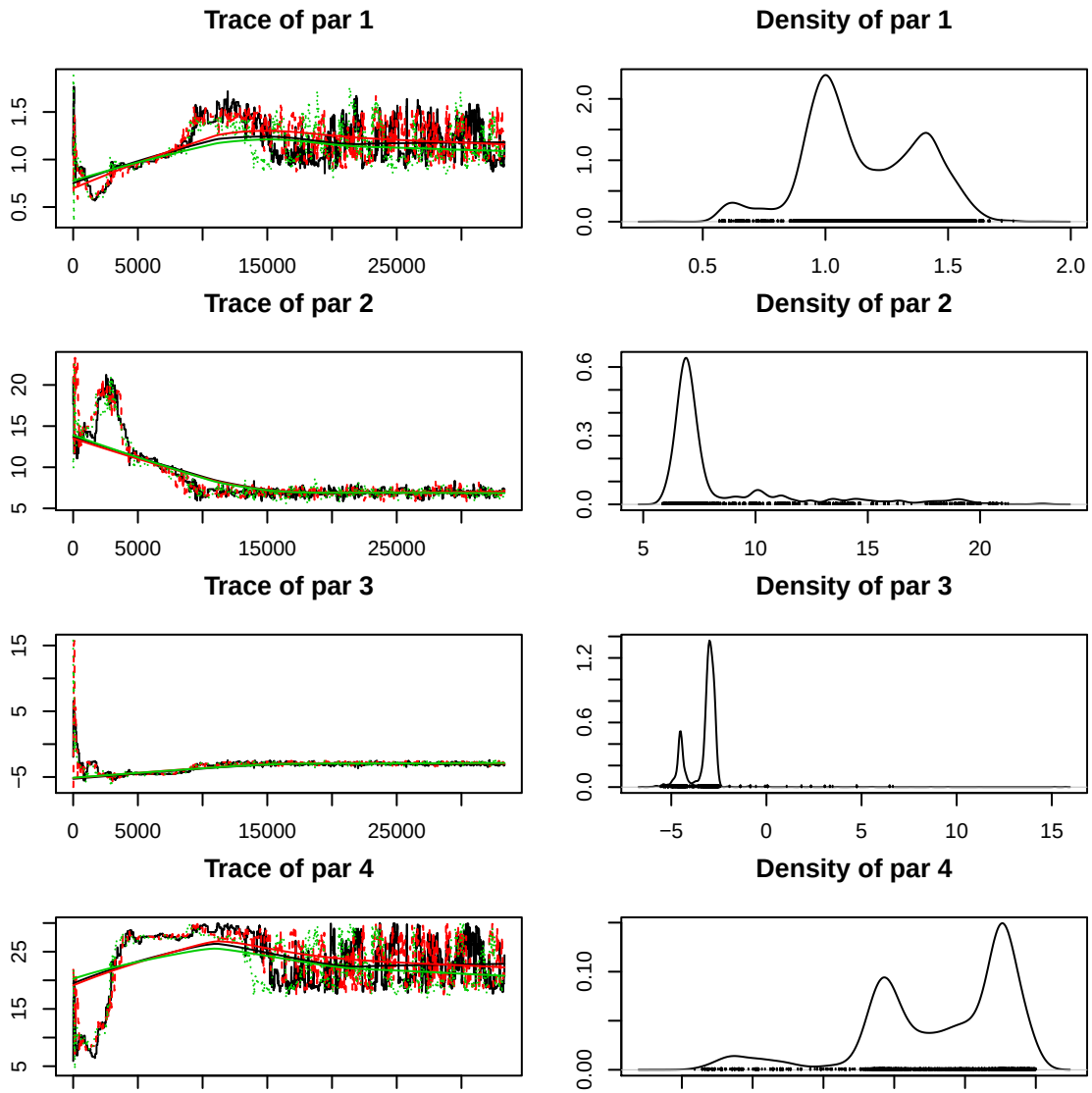
```
Parameter values  1.14777233492726 8.58034461782013 -3.26311226684232 23.1648750942292 -0.0747293151574114 0.059
```

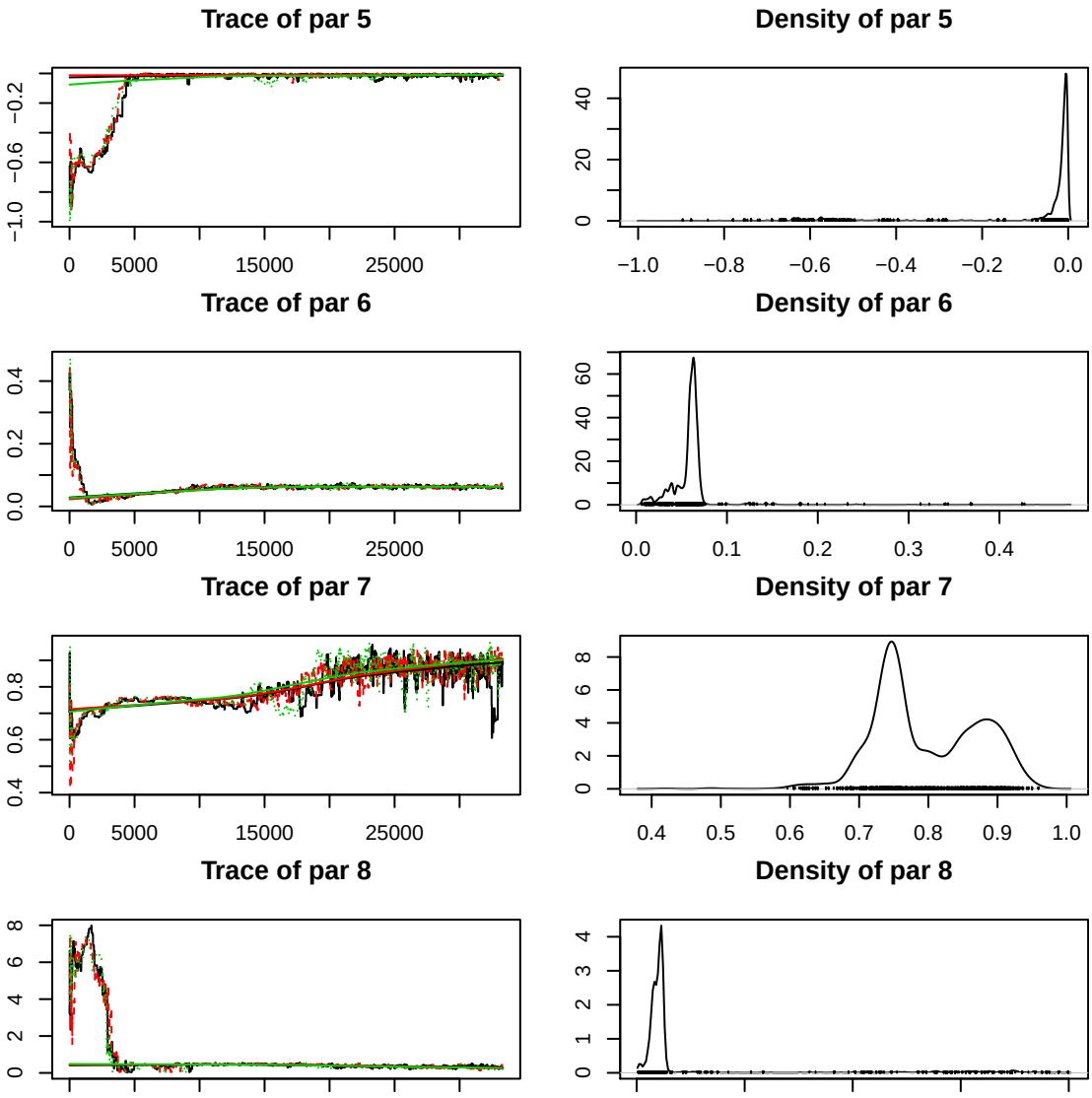
```
Problem encountered in the calculation of the likelihood with parameter 1.147772334927268.58034461782013-3.26311
 Error message wasError in PRELES(PAR = PAR, TAir = TAir, VPD = VPD, Precip = Precip, CO2 = CO2, : could not fin

 set result of the parameter evaluation to -Inf ParameterValues
```

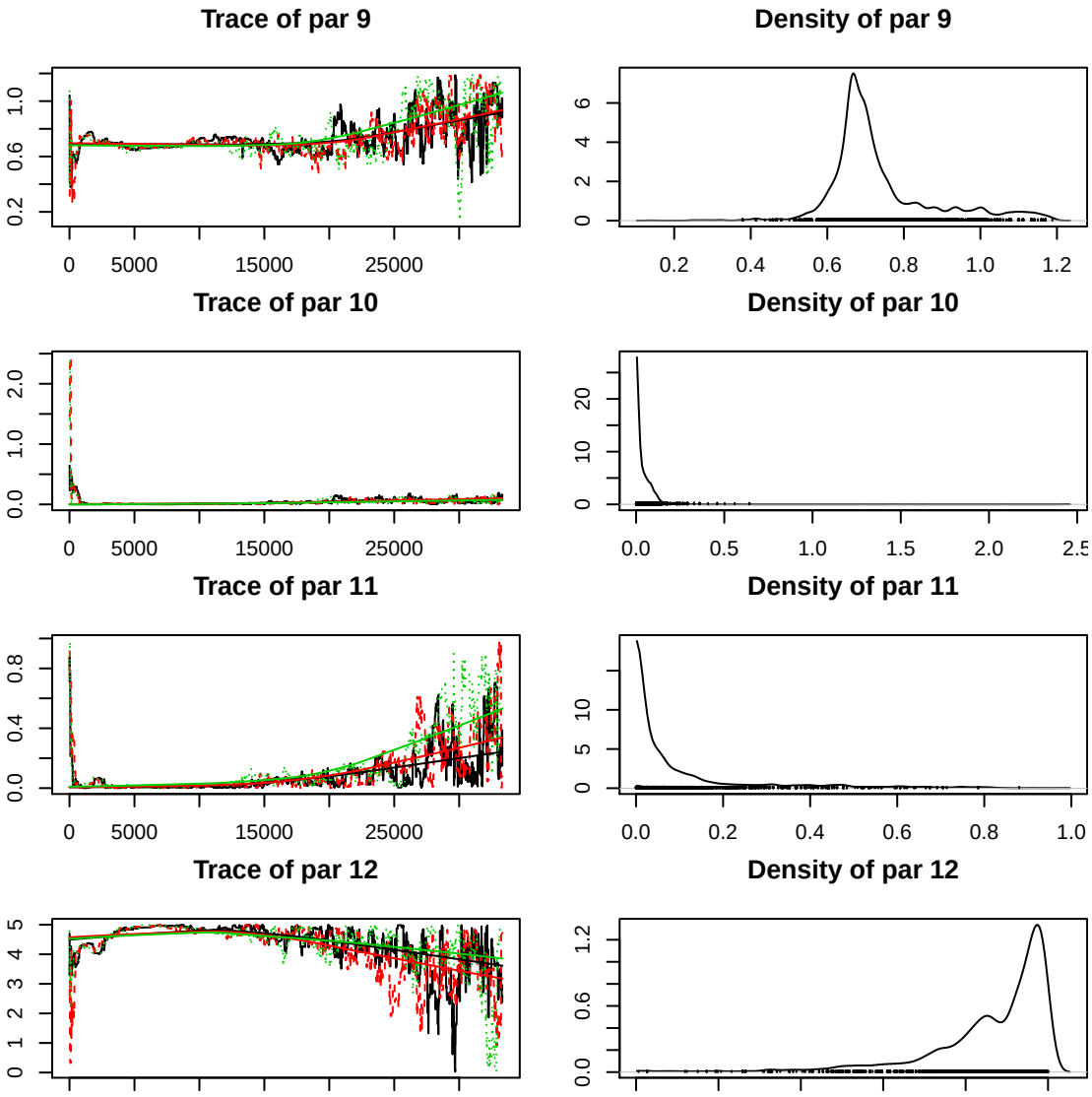
```
# # # # # # # # # # # # # # # # # # # # # # # # # # # #
## MCMC chain summary ##
# # # # # # # # # # # # # # # # # # # # # # # # # # # #

# MCMC sampler:  DEzs
# Nr. Chains:  3
# Iterations per chain:  33334
# Rejection rate:  0.957
# Effective sample size:  161
# Runtime:  115.331  sec.

# Parameters
         psf    MAP   2.5% median  97.5%
par 1  1.016  1.009  0.639  1.094  1.562
par 2  1.013  6.868  6.180  7.101 18.906
par 3  1.040 -3.071 -4.862 -3.033 -2.567
par 4  1.018 19.301  8.658 24.638 29.411
par 5  1.023 -0.007 -0.629 -0.011 -0.001
par 6  1.019  0.064  0.016  0.061  0.092
par 7  1.036  0.834  0.676  0.768  0.928
par 8  1.012  0.284  0.127  0.416  6.705
par 9  1.023  1.075  0.558  0.695  1.118
par 10 1.010  0.081  0.000  0.021  0.145
par 11 1.111  0.221  0.001  0.041  0.663
par 12 1.023  4.724  2.134  4.597  4.986
par 13 1.012  0.129  0.120  0.135  0.190
par 14 1.006  0.163  0.150  0.168  1.092

## DIC:  -Inf
## Convergence
 Gelman Rubin multivariate psrf:

## Correlations
        par 1  par 2  par 3  par 4  par 5  par 6  par 7  par 8  par 9
par 1   1.000 -0.598  0.307  0.738  0.540  0.261  0.172 -0.515 -0.017
par 2  -0.598  1.000 -0.326 -0.396 -0.836 -0.168 -0.544  0.699 -0.207
par 3   0.307 -0.326  1.000 -0.156  0.002  0.663  0.143  0.020  0.129
par 4   0.738 -0.396 -0.156  1.000  0.676 -0.107  0.082 -0.719 -0.133
par 5   0.540 -0.836  0.002  0.676  1.000 -0.102  0.503 -0.937  0.132
par 6   0.261 -0.168  0.663 -0.107 -0.102  1.000  0.010  0.035 -0.040
par 7   0.172 -0.544  0.143  0.082  0.503  0.010  1.000 -0.478  0.324
par 8  -0.515  0.699  0.020 -0.719 -0.937  0.035 -0.478  1.000 -0.106
par 9  -0.017 -0.207  0.129 -0.133  0.132 -0.040  0.324 -0.106  1.000
par 10 -0.031  0.065  0.670 -0.209 -0.177  0.539  0.060  0.160  0.153
par 11  0.001 -0.205  0.319 -0.163  0.112  0.223  0.473 -0.124  0.626
par 12  0.018  0.068 -0.322  0.247  0.072 -0.197 -0.411 -0.064 -0.420
par 13 -0.028  0.235  0.612 -0.133 -0.308  0.665 -0.163  0.234 -0.061
par 14 -0.157  0.422  0.500 -0.369 -0.621  0.727 -0.365  0.510 -0.162
       par 10 par 11 par 12 par 13 par 14
par 1  -0.031  0.001  0.018 -0.028 -0.157
par 2   0.065 -0.205  0.068  0.235  0.422
par 3   0.670  0.319 -0.322  0.612  0.500
par 4  -0.209 -0.163  0.247 -0.133 -0.369
```

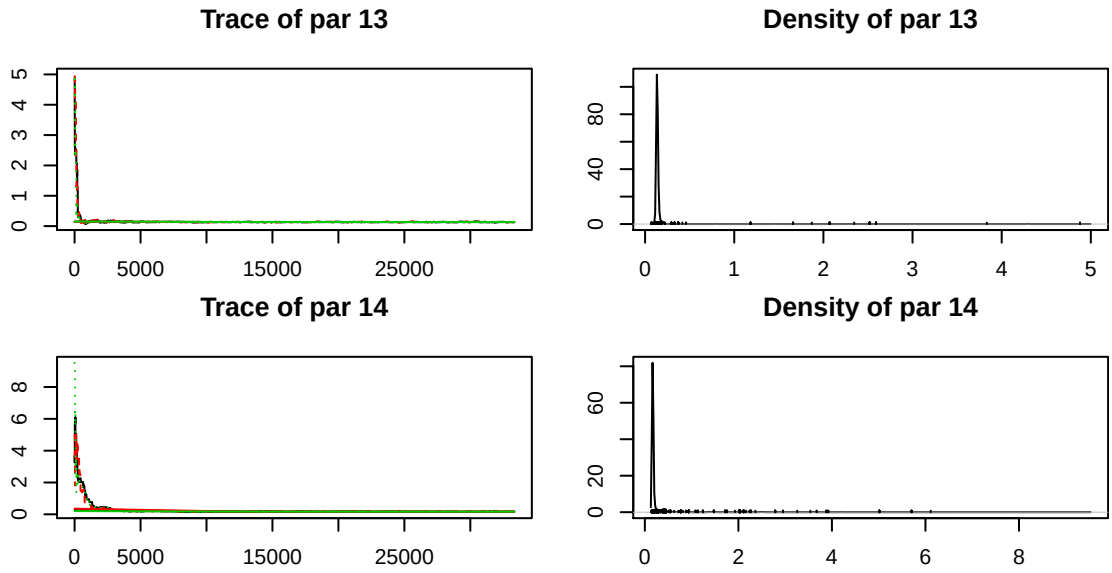
```
par 5  -0.177  0.112  0.072 -0.308 -0.621
par 6   0.539  0.223 -0.197  0.665  0.727
par 7   0.060  0.473 -0.411 -0.163 -0.365
par 8   0.160 -0.124 -0.064  0.234  0.510
par 9   0.153  0.626 -0.420 -0.061 -0.162
par 10  1.000  0.285 -0.347  0.783  0.584
par 11  0.285  1.000 -0.666  0.172  0.078
par 12 -0.347 -0.666  1.000 -0.144 -0.157
par 13  0.783  0.172 -0.144  1.000  0.739
par 14  0.584  0.078 -0.157  0.739  1.000
```

```
par(mar = c(1, 4, 4, 0))
plot(fit2)
```




Trace of par 5
Density of par 5
Trace of par 6
Density of par 6
Trace of par 7
Density of par 7
Trace of par 8
Density of par 8


Trace of par 9
Density of par 9
Trace of par 10
Density of par 10
Trace of par 11
Density of par 11
Trace of par 12
Density of par 12

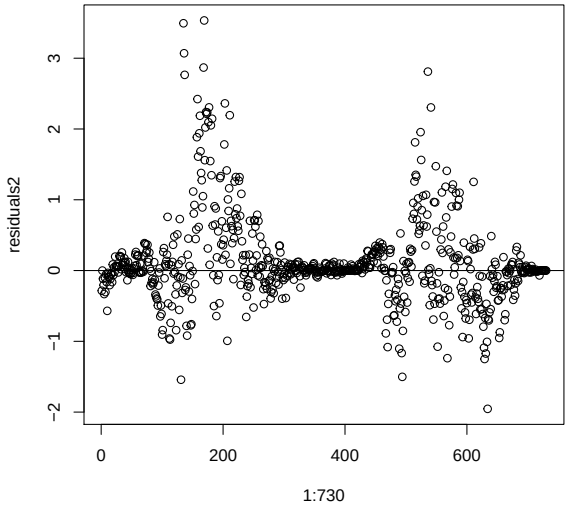


The convergence criteria look almost alright.

```
parSamples2 ← getSample(fit2, thin = 100)
getPreds2 ← function(pars) {
    thispar ← par$def
    thispar[pars2tune2] ← pars
    preds ← with(s1, PRELES(PAR = PAR, TAir = TAir, VPD = VPD, Precip = Precip,
        CO2 = CO2, fAPAR = fAPAR, p = thispar)$GPP)
    return(preds)
}
# now run prediction with each parameter combination
preds2 ← sapply(1:nrow(parSamples2), function(x) getPreds2(parSamples2[x,
    ]))
RMSE(rowMeans(preds2), GPPobs)
```

```
[1] 0.636734
```

```
residuals2 ← GPPobs − rowMeans(preds2)
plot(1:730, residuals2)
abline(h = 0)
```



Okay, this was not a break-through. The RMSE is actually higher than in the first fit, and the residuals don't look any better (we didn't expect them to, only the likelihood now allows for higher variances).

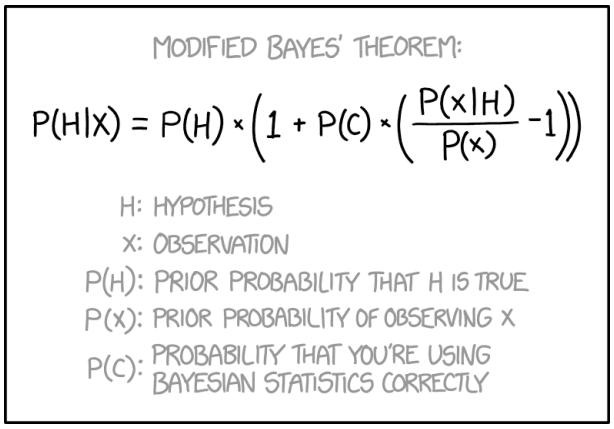
We can now go on to try alternative likelihoods (e.g. Laplace = double exponentials, rather than normal, which would allow less variation around the mean), possibly add lag effects (i.e. temporal autocorrelation in the variance term), etc.

6 *Conclusion*

With our model becoming more complicated than a regression, new challenges may arise. Here we have mainly seen that (a) more efficient samplers are called for; (b) everything just takes more time (despite (a)); and (c) the likelihood requires some craftmanship.

Things we haven't seen here are correlation between parameters, which may lead to unidentifiability (= equifinality). They would require a reformulation of the parameters, e.g. making one a function of the other before fitting.

This was just the start. When you want to do your own analysis, you should familiarise yourself more with the concepts at hand, for which I can only recommend Dietze (2017). I hope you now have something to play with.



Bibliography

Dietze, M. C. (2017). *Ecological Forecasting*. Princeton University Press, Princeton, N.J.

Lemoine, N. P. (2019). Moving beyond noninformative priors: why and how to choose weakly informative priors in Bayesian analyses. *Oikos*, 128(7):912–928.

Minunno, F., Peltoniemi, M., Launiainen, S., Aurela, M., Lindroth, A., Lohila, A., Mammarella, I., Minkkinen, K., and Mäkelä, A. (2016). Calibration and validation of a semi-empirical flux ecosystem model for coniferous forests in the Boreal region. *Ecological Modelling*, 341:37–52.

Peltoniemi, M., Markkanen, T., and Härkönen, S. (2015). Consistent estimates of gross primary production of Finnish forests - comparison of estimates of two process models. *Boreal Environment Research*, 20:196–212.