Synthetic Likelihood Lab

We will use the package `synlik` by Matteo Fasiolo and Simon Wood (2015). The package is used to do inference on models where the likelihood is unavailable. To be able to fit the model, it must be possible to simulate data from it.

First, install the package:

```
install.packages("synlik")
```

```
library("synlik")
source("plot_smcmc.R")
```

The main function `synlik()` creates an object of class `synlik` which consists of a *simulator* function, which contains instructions on how to simulate the data from the specified model, and a *summaries* function which is used to calculated the desired summary statistics using the simulated data.

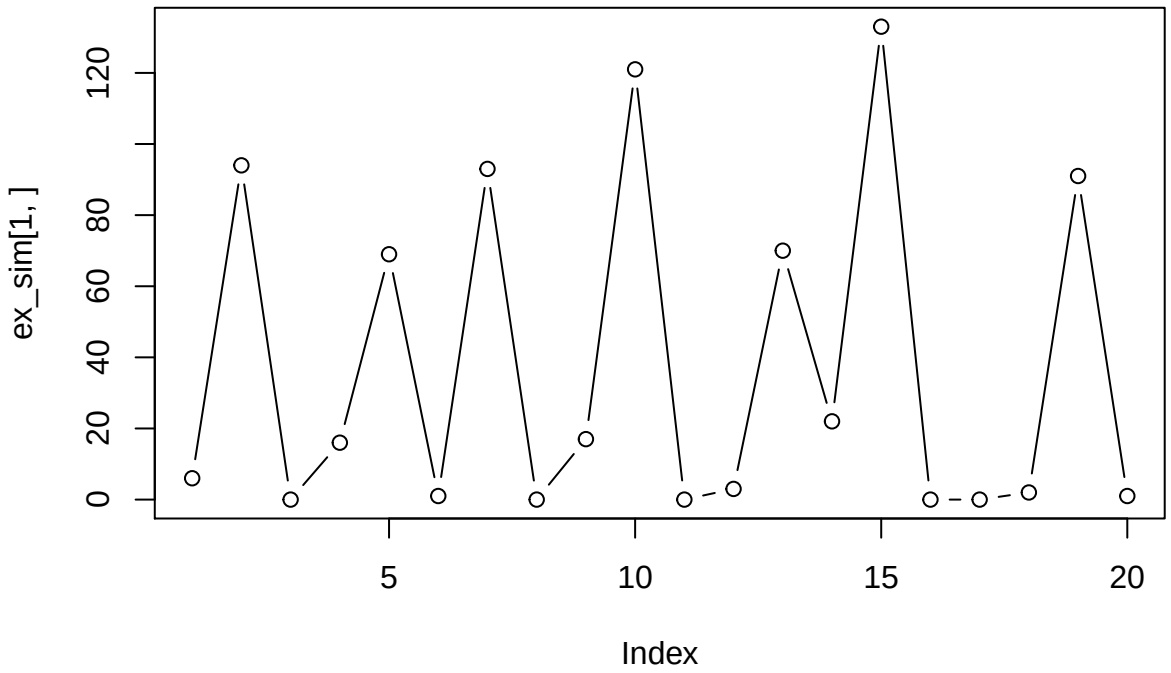
We will go through two examples that come with the `synlik` package.

The Ricker model

The model is $N_t = rN_{t-1} \exp -N_{t-1} + \epsilon_t$ where $\epsilon_t \sim N(0, \sigma_e^2)$ are the independent process noise terms and r is the growth rate parameter controlling the model dynamics. The observations are given by $Y_t \sim \text{Pois}(\phi N_t)$. The goal is to make inference about $\theta^{\mathbf{T}} = (\mathbf{r}, \sigma^{\mathbf{2}}, \phi)$.

The simulation for this model is already contained within `rickerSimul()`, to see how it works try one simulation with $log(r) = 3.8, \sigma = 0.3, \phi = 10$

```
set.seed(1234)
rParams <- c(logR = 3.8, logSigma = log(0.3), logPhi = log(10) )
ex_sim <- rickerSimul(rParams,
                      nsim=1,
                      extraArgs = list(nObs = 20, nBurn = 0))
plot(ex_sim[1,],type="b")
```



As described in the Wood (2010) paper, the summary statistics that were chosen for the Ricker model are

- the autocovariances to lag 5
- the coefficients of the cubic regression of the ordered differences $y_t - y_{t-1}$ on their observed values
- the coefficients β_1, β_2 of the autoregression $y_{t+1}^{0.3} = \beta_1 y_t^{0.3} + \beta_2 y_t^{0.6} + \epsilon_t$
- the mean population
- the number of zeros observed

These statistics are contained within `rickerStats()`. This function is designed to be used within the `synlik()` function, but we can see the code for the summary statistics:

```
rickerStats
```

```
## function (x, extraArgs, ...)
## {
##     obsData <- as.vector(extraArgs$obsData)
##     stopifnot(length(obsData) != 0)
##     if (!is.matrix(x))
##         x <- matrix(x, 1, length(x))
##     tx <- t(x)
##     X0 <- t(orderDist(tx, obsData, np = 3, diff = 1))
##     X0 <- cbind(X0, t(nlar(tx^0.3, lag = c(1, 1), power = c(1,
##         2))))
##     X0 <- cbind(X0, rowMeans(x), rowSums(x == 0))
##     X0 <- cbind(X0, t(slAcf(tx, max.lag = 5)))
##     X0
## }
## <environment: namespace:synlik>
```

Lets create the `synlik` object:

```
ricker_sl <- synlik(simulator = rickerSimul,
                    summaries = rickerStats,
                    param = rParams,
                    extraArgs = list("nObs" = 50, "nBurn" = 50))
```

Check that is it is the right class

```
class(ricker_sl)
```

```
## [1] "synlik"
## attr(,"package")
## [1] "synlik"
```

Now simulate the data (a single data set)

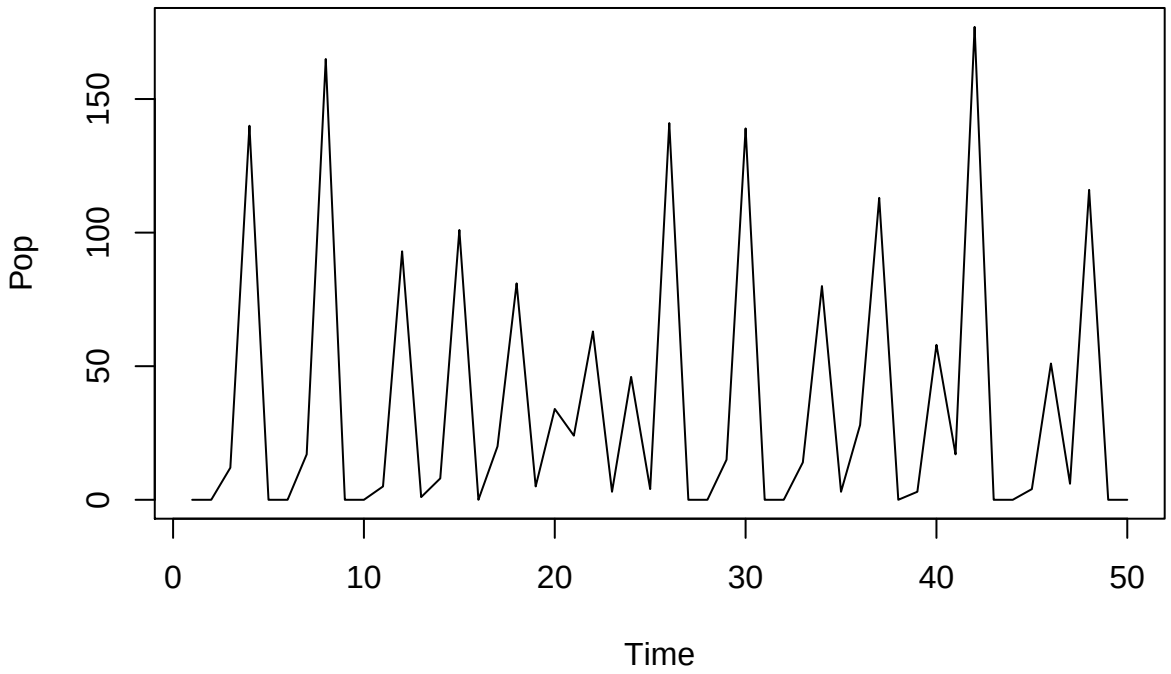
```
ricker_sl@data <- simulate(ricker_sl, nsim = 1, seed = 54)
```

We could quickly view the data using

```
plot(as.vector(ricker_sl@data),type="l")
```

But it is suggested to add a (slightly prettier) plotting function to the object:

```
ricker_sl@plotFun <- function(input, ...) {
    plot(drop(input), type = 'l', ylab = "Pop",
         xlab = "Time", ...)
    }
plot(ricker_sl)
```

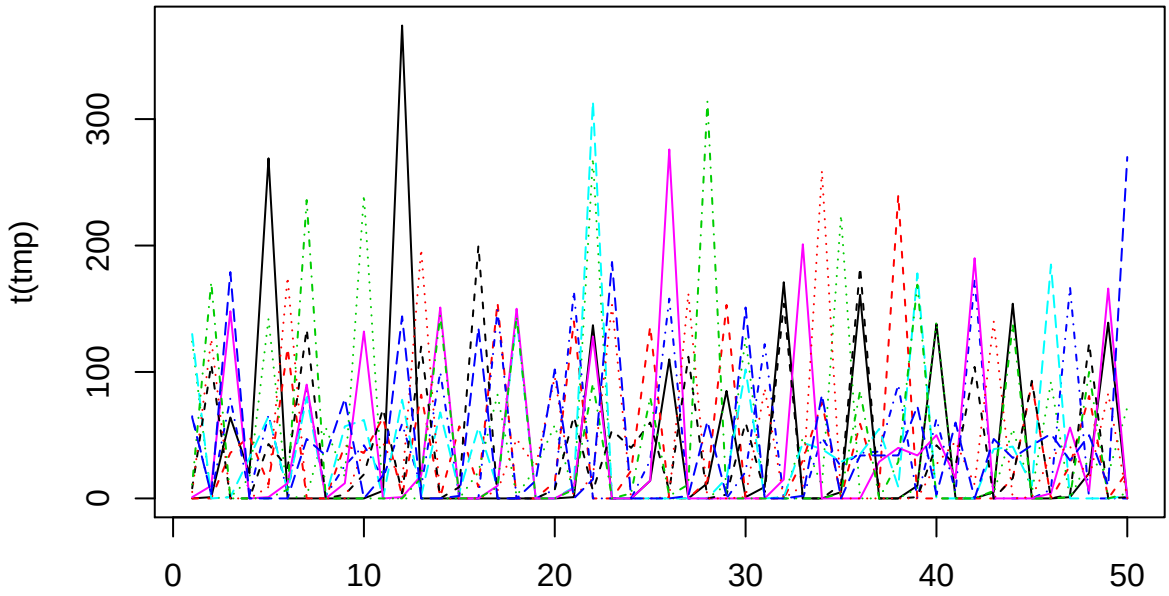


Simulate several data sets

```
tmp <- simulate(ricker_sl, nsim = 10)
dim(tmp)
```

```
## [1] 10 50
```

```
matplot(t(tmp),type="l")
```



Now we need to store the simulated (reference) data in the object, so that its summary statistics can be used as "observed" summary statistics as compared to simulated summary statistics

```
ricker_sl@extraArgs$obsData <- ricker_sl@data
```

Then we can simulate the summary statistics using `stats=TRUE`

```
(tmp <- simulate(ricker_sl, nsim = 2, stats = TRUE))
```

```
##           [,1]         [,2]          [,3]       [,4]       [,5]  [,6] [,7]
## [1,] 0.5255538 0.0006513724  3.377820e-05  0.1526005 -0.2090391 35.44   19
## [2,] 0.9574071 0.0001031115 -1.446878e-08 -0.2195997 -0.2439037 36.02   14
##          [,8]      [,9]     [,10]      [,11]     [,12]      [,13]
## [1,] 3544.366 -894.8513 -255.0406 -493.98597 215.6840 -774.70329
## [2,] 2235.700 -896.9890 -322.3338  -71.68088 314.0995  -83.26227
```

Compare these to the summary statistics for the "observed" data

```
rickerStats(ricker_sl@data,ricker_sl@extraArgs)
```

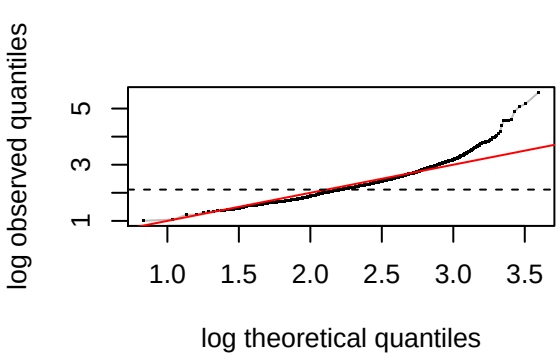
```
##      [,1]          [,2]          [,3]        [,4]       [,5]  [,6] [,7]
## [1,]    1 -1.234609e-19 -1.056699e-20 -0.02226859 -0.2529841 35.74   16
##          [,8]      [,9]     [,10]     [,11]     [,12]     [,13]
## [1,] 2556.872 -849.3822 -619.8141 -329.6107 925.8476 -407.3248
```

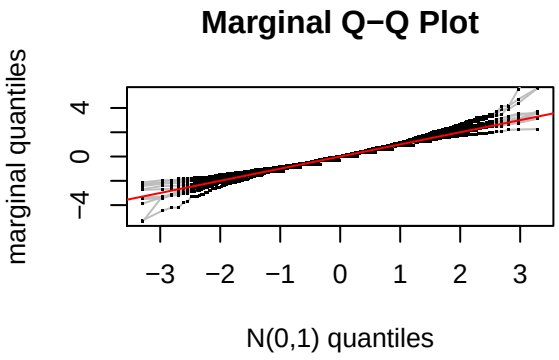
Notice how the first entry is 1; this is because this statistic is a coefficient from a regression of an ordered difference series on itself. So the simulated statistics are statistics calculated from new sets of simulated data, and compared with the "observed" data.

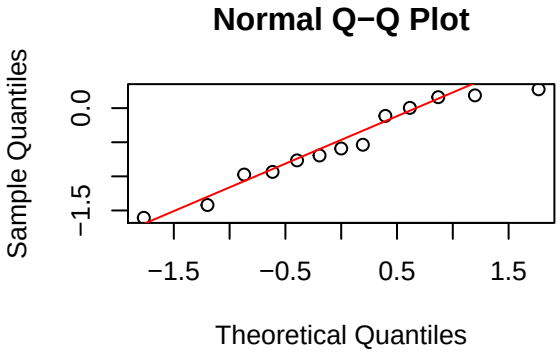
We want to check the normality of the summary statistics, because this is an assumption in the synthetic likelihood methods. The package has a built in function `checkNorm()` to check whether the distribution of the random summary statistics is multivariate normal. The default is 1000 simulations. The function reports the proportion of data with substantial deviation from the ideal line.

```
checkNorm(ricker_sl)
```

```
##
## proportion |log(z)-log(q)|>.25 =  0.07
```








```
##  [1] 0.720 0.063 0.139 0.289 0.086 0.093 0.174 0.068 0.891 0.582 0.392
## [12] 0.929 0.236
```

If we are satisfied with normality we can now explore the synthetic likelihood.

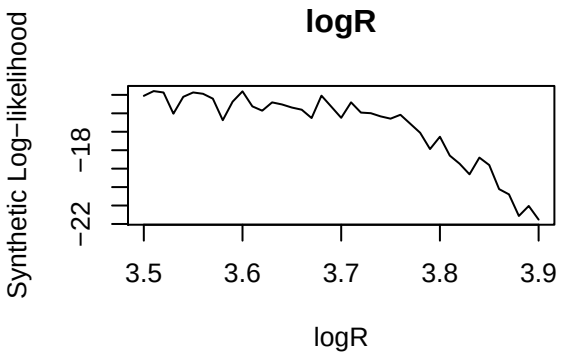
We can use `slik()` to estimate the value of the synthetic likelihood at particular parameters:

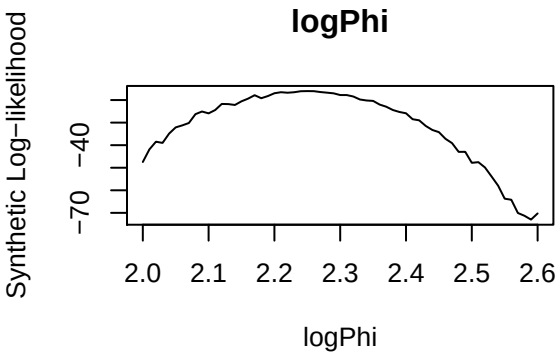
```
slik(ricker_sl,
     param = rParams,
     nsim = 1000)
```

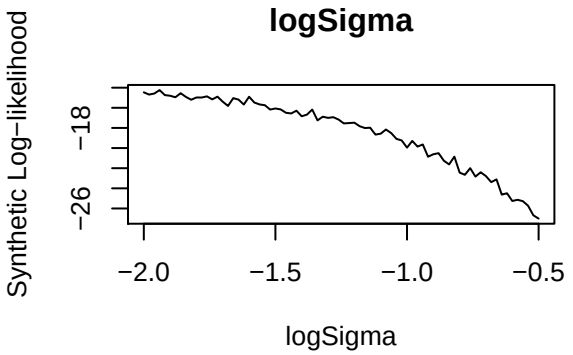
```
## [1] -17.56645
```

We can also look at slices with respect to each parameter at a time, while holding the other fixed

```
slice(object = ricker_sl,
      ranges = list("logR" = seq(3.5, 3.9, by = 0.01),
      "logPhi" = seq(2, 2.6, by = 0.01),
      "logSigma" = seq(-2, -0.5, by = 0.02)),
      param = rParams,
      nsim = 1000)
```

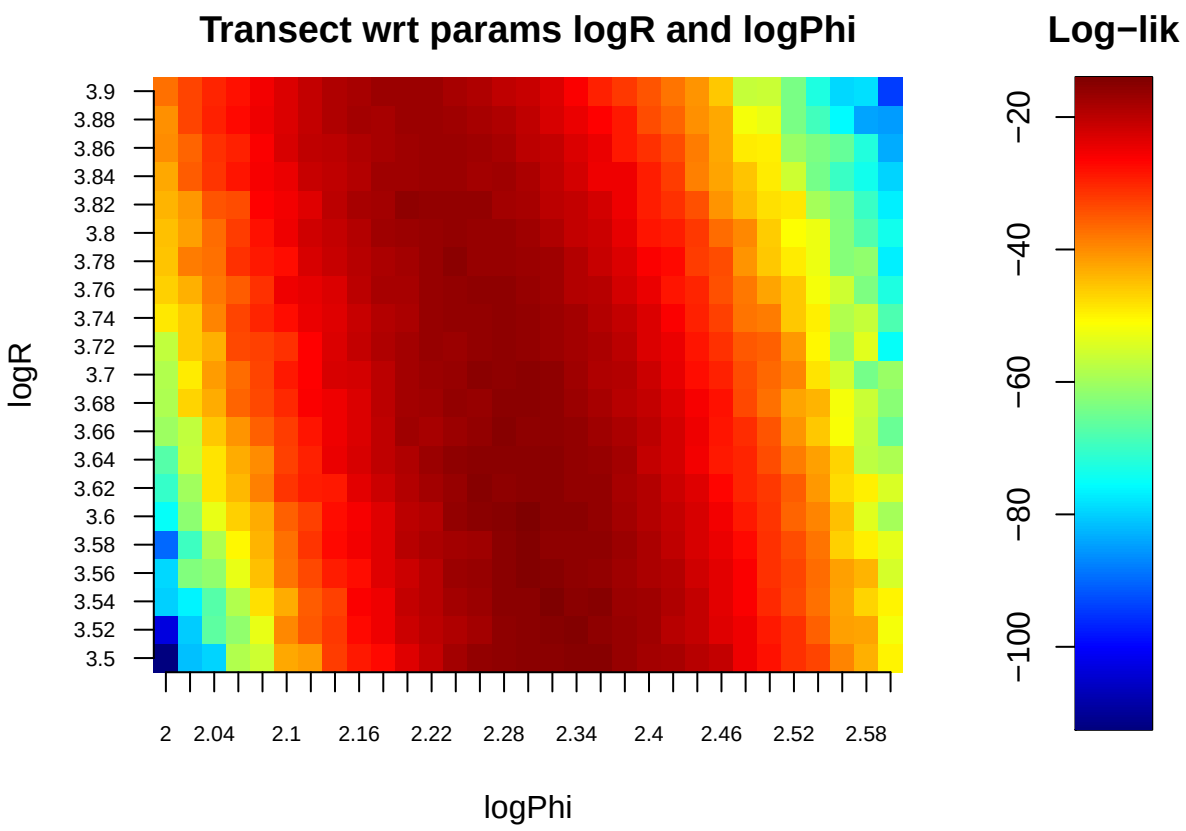






A 2-D slice can be more useful sometimes (this is somewhat computationally intensive)

```
slice(object = ricker_sl,
      ranges = list("logR" = seq(3.5, 3.9, by = 0.02),
      "logPhi" = seq(2, 2.6, by = 0.02)),
      pairs = TRUE,
      param = rParams,
      nsim = 1000,
      multicore = TRUE,
      ncores = 2)
```



Now that we feel pretty confident about our "initial guess" of the parameter $\theta^{\mathbf{T}}$ we can start using MCMC to estimate the parameters ...

```
ricker_sl <- smcmc(ricker_sl,
                   initPar = c(3.2, -1, 2.6),
                   niter = 10,
                   burn = 3,
                   priorFun = function(input, ...) sum(input),
                   propCov = diag(c(0.1, 0.1, 0.1))^2,
                   nsim = 500)
```

Notice the class of the object has now changed but we haven't lost any of the previously stored information.

```
class(ricker_sl)
```

```
## [1] "smcmc"
## attr(,"package")
## [1] "synlik"
```

You can print the entire object, too (not shown – there's a lot in there!)

```
ricker_sl
```

View the chains using `ricker_sl@chains`:

We can continue the MCMC

```
ricker_sl <- continue(ricker_sl, niter = 10)
```

So that we don't have to compute very many iterations ourselves, we can use the 20,000 pre-computed Ricker MCMC chains that comes with the package

```
data("ricker_smcmc")
addline1 <- function(parNam, ...)
    abline(h = ricker_smcmc@param[parNam], lwd = 2, lty = 2, col = 3)
addline2 <- function(parNam, ...)
    abline(v = ricker_smcmc@param[parNam], lwd = 2, lty = 2, col = 3)
```

The blowfly example

The proposed model for the blowfly example is $N_t = R_t + S_t$ where $R_t \sim Pois(PN_{t-\tau}e^{-\frac{N_{t-\tau}}{N_0}}e_t)$ represents the reproduction process, and $S_t \sim binom(e^{-\delta\epsilon_t}, N_{t-1})$, represents the adult survival. $e_t and \epsilon_t$ are independent gamma random variables with means and variances equal to σ_p^2 and σ_d^2 repectively.

The model is already contained within `blowSimul` and the summary statistics function is `blowStats`. The summary statistics were chosen to be - autocovariances to lag 11 - the coefficients of the cubic regression of the ordered differences $y_t - y_{t-1}$ on their observed values - mean(N_t) - mean(N_t)-median(N_t) - the number of turning points observed - the coefficients of the autoregression $N_i = \beta_0 N_{i-12} + \beta_1 N_{i-12}^2 + \beta_2 N_{i-12}^3 + \beta_3 N_{i-2} + \beta_4 N_{i-2}^2 + \epsilon_i$

We can create the `synlik` model using

```
bParams <- c( delta = 0.16, P = 6.5, N0 = 400,
              var.p = 0.1, tau = 14, var.d = 0.1)
blow_sl <- synlik(simulator = blowSimul,
                  summaries = blowStats,
                  param = bParams,
                  extraArgs = list("nObs" = 200, "nBurn" = 200, "steps" = 1),
                  plotFun = function(input, ...){
                      plot(drop(input), type = 'l', ylab = "Pop", xlab =
                               "Time", ...)
                  }
)
```

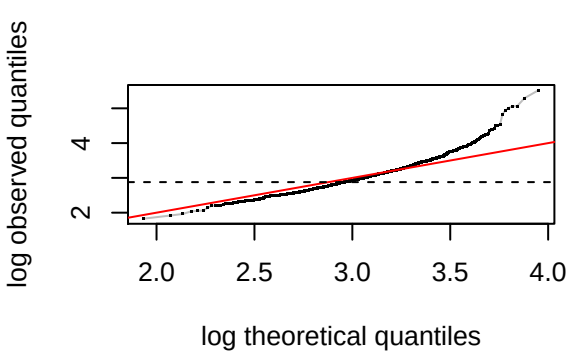
Again, we simulate the data and store it:

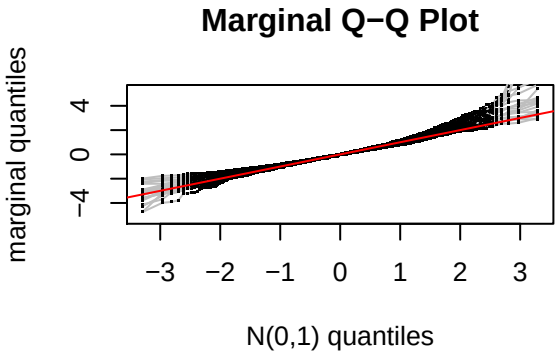
```
blow_sl@data <- simulate(blow_sl, seed = 84)
blow_sl@extraArgs$obsData <- blow_sl@data
```

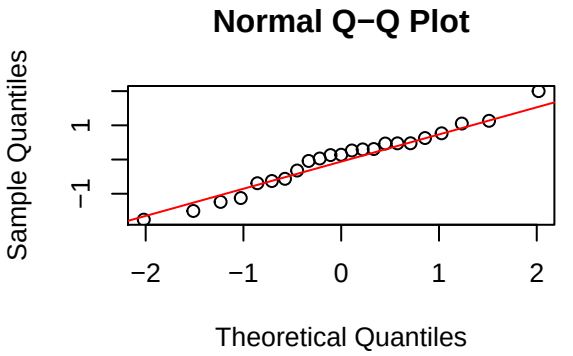
Check the normality of the summary statistics

```
checkNorm(blow_sl)
```

```
##
## proportion |log(z)-log(q)|>.25 =  0.082
```








```
##  [1] 0.032 0.330 0.827 0.963 0.840 0.077 0.799 0.342 0.151 0.350 0.181
## [12] 0.382 0.612 0.891 0.951 0.921 0.939 0.964 0.918 0.938 0.872 0.480
## [23] 0.721
```

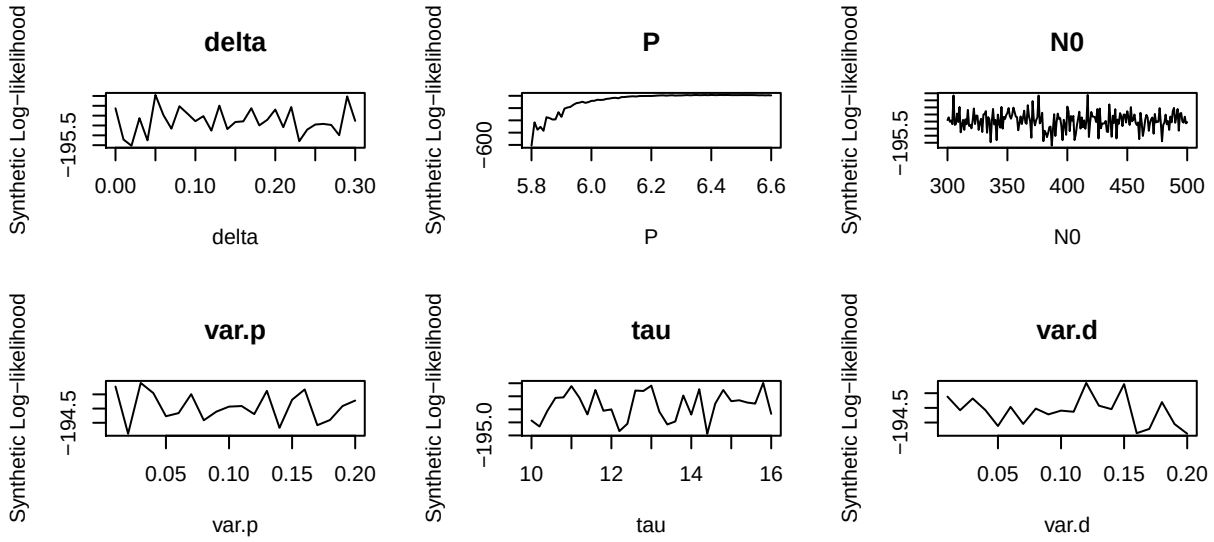
Take a look at the estimated value of the synthetic likelihood

```
slik(blow_sl,
     param = bParams,
     nsim = 1000)
```

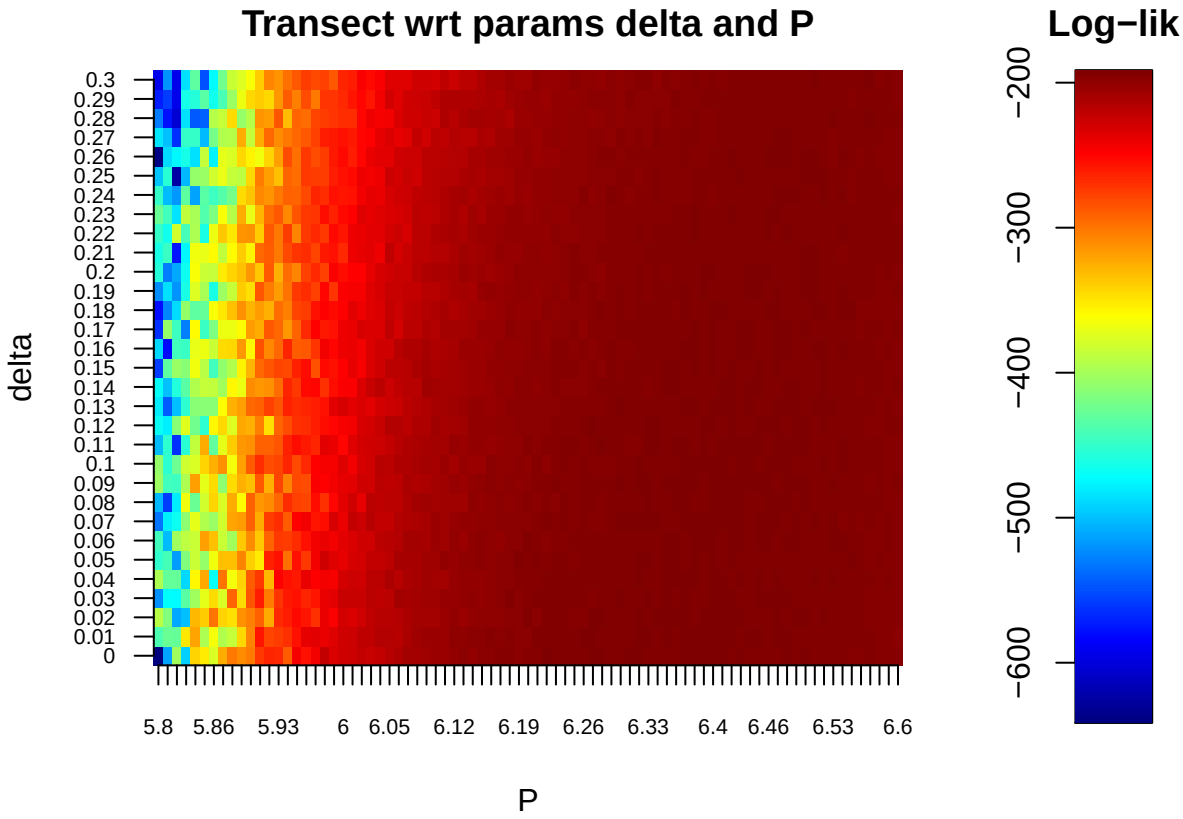
```
## [1] -193.4871
```

Take a look at a slice (computationally intensive):

```
slice(object = blow_sl,
      ranges = list(delta = seq(0, 0.3, by = 0.01),
      P = seq(5.8, 6.6, by = 0.01),
      N0 = seq(300, 500, by = 1),
      var.p = seq(0.01, 0.2, by = 0.01),
      tau = seq(10, 16, by = 0.2),
      var.d = seq(0.01, 0.2, by = 0.01)),
      param = bParams,
      nsim = 1000)
```




```
slice(object = blow_sl,
    ranges = list(delta = seq(0, 0.3, by = 0.01),
        P = seq(5.8, 6.6, by = 0.01)),
      pairs = TRUE,
      param = bParams,
      nsim = 1000,
      multicore = TRUE,
      ncores = 2)
```



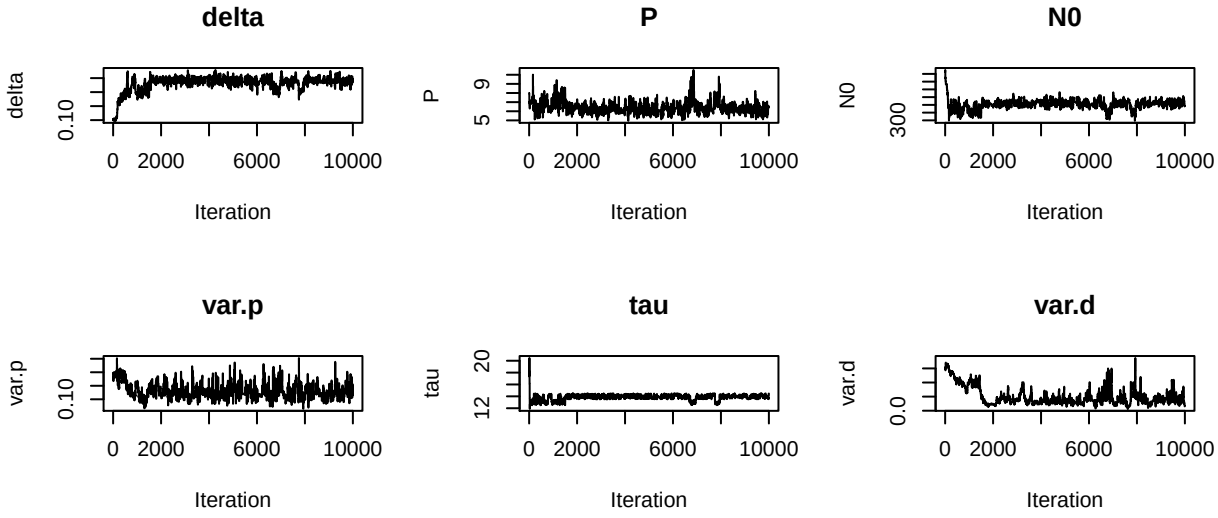
The MCMC chains:

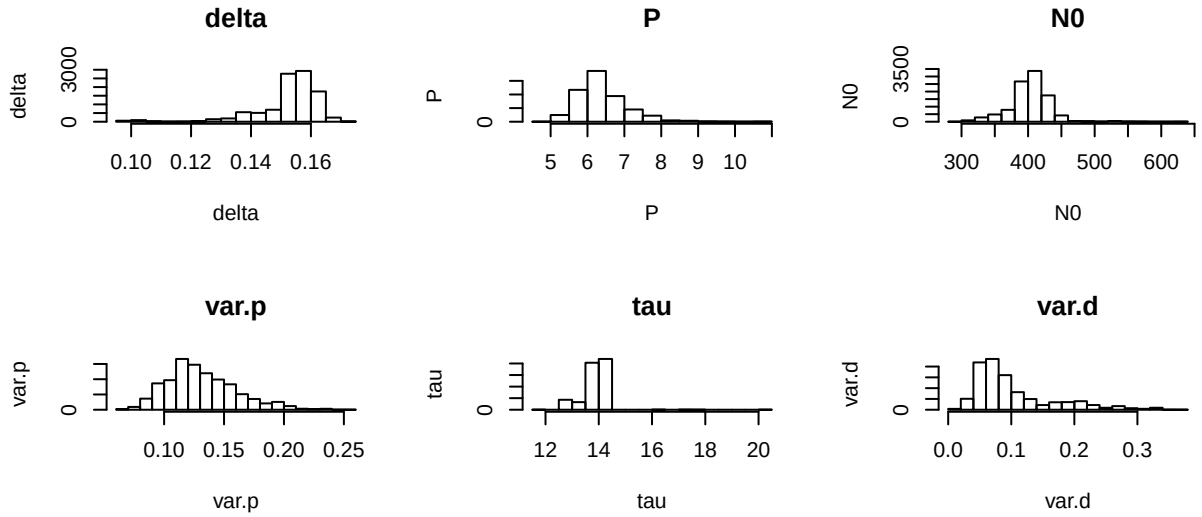
```
blow_sl <- smcmc(blow_sl,
        initPar = log(bParams),
        niter = 2,
        burn = 0,
        propCov = diag(rep(0.001, 6)),
        nsim = 500,
        prior = function(input, ...){
                    sum(input) +
                        dunif(input[4], log(0.01), log(1), log = TRUE) +
                            dunif(input[6], log(0.01), log(1), log = TRUE)
        },
        targetRate = 0.15,
        multicore = FALSE
)
```

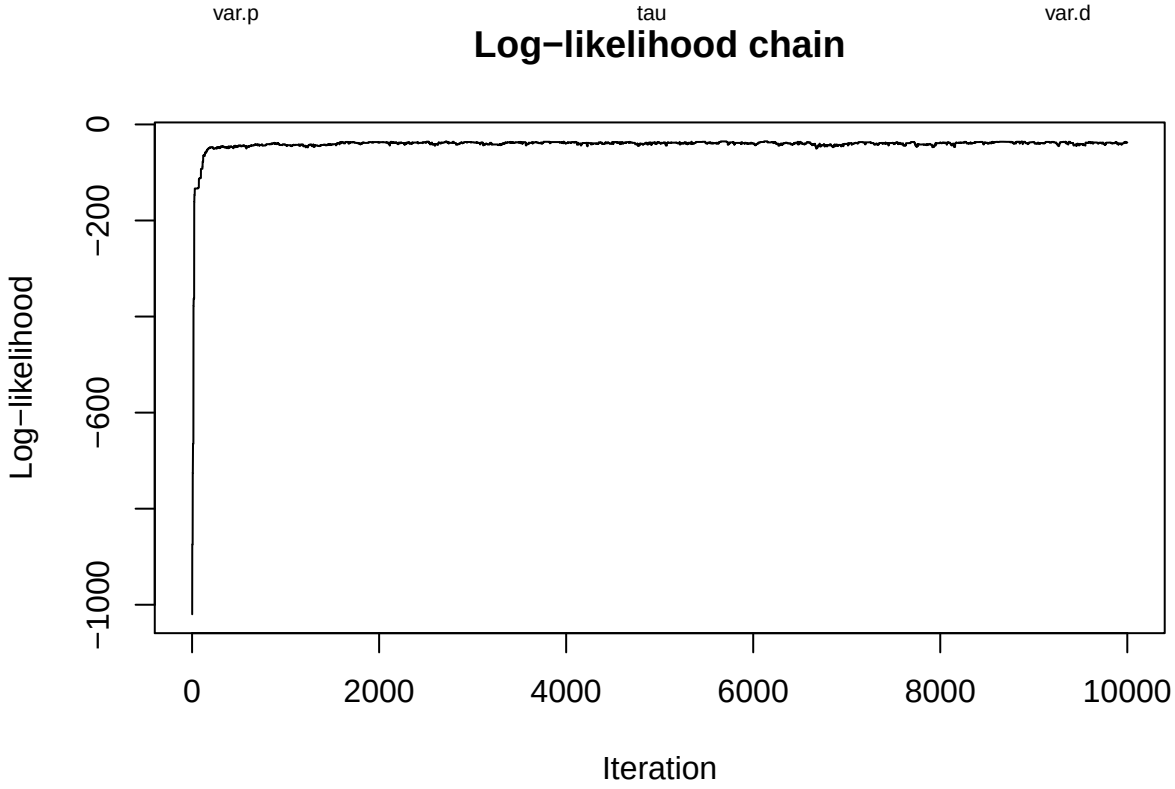
Again, instead of trying to run 20,000 interations we will use the precomputed data from the package. We can plot these results on the original scale

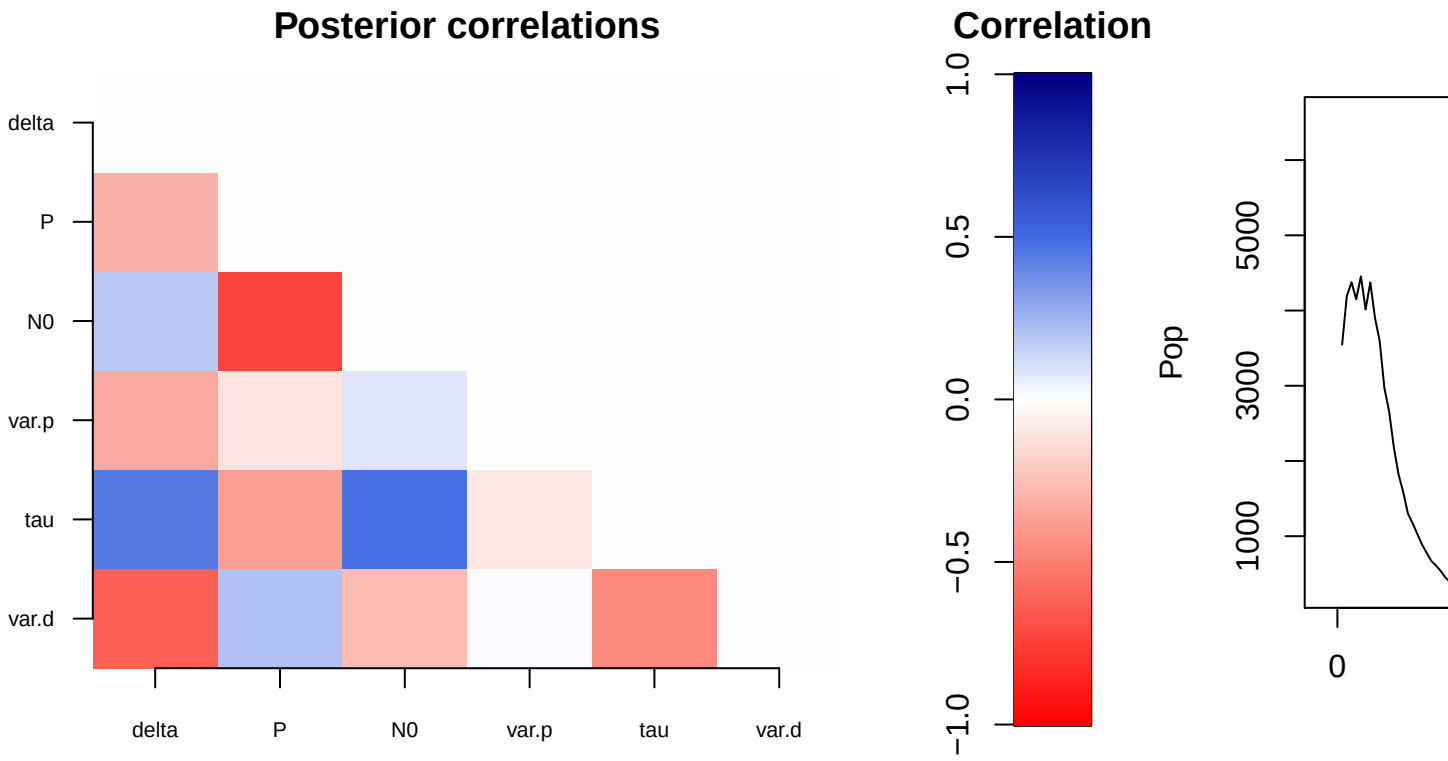
```
par(ask=FALSE)
data(blow_smcmc)
tmpTrans <- rep("exp", 6)
names(tmpTrans) <- names(blow_smcmc@param)
plot(blow_smcmc, trans = tmpTrans)
```

```
## modified .plot.smcmc
```




delta
P
N0
var.p
tau
var.d
0.10 0.12 0.14 0.16
5 6 7 8 9 10
300 400 500 600
0.10 0.15 0.20 0.25
12 14 16 18 20
0.0 0.1 0.2 0.3


Log−likelihood chain
Log−likelihood
−1000
−600
−200
0
0
2000
4000
6000
8000
10000
Iteration

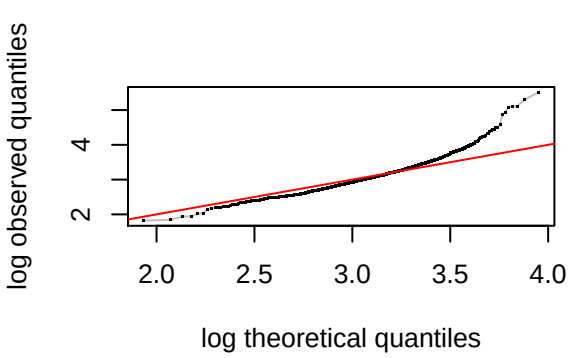


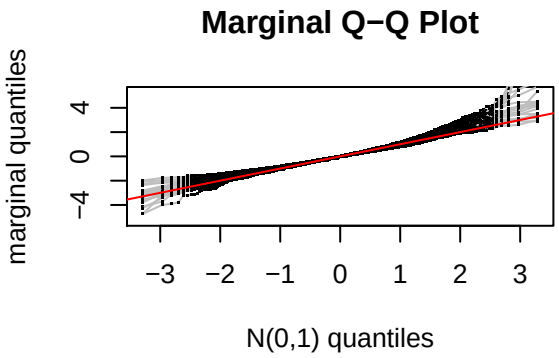
Now we can try using some experimental data from the Nicholson (1954) blowfly experiments.

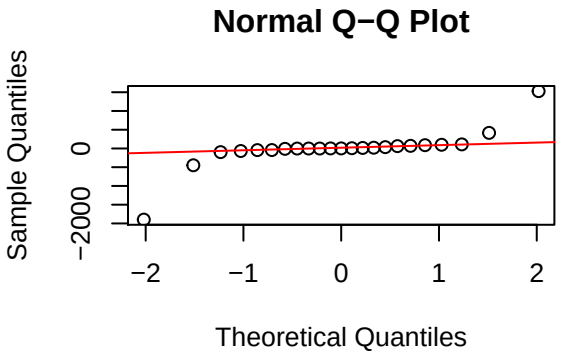
```
data(bf1)
blow_sl@data <- bf1$pop
blow_sl@extraArgs$obsData <- blow_sl@data
```

```
par(ask=FALSE)
checkNorm(blow_sl)
```

```
##
## proportion |log(z)-log(q)|>.25 =  0.079
```








```
##  [1] 0.000 0.238 0.963 0.023 1.000 0.000 1.000 0.000 0.000 0.000 0.000
## [12] 0.000 0.022 0.193 0.384 0.508 0.541 0.553 0.536 0.554 0.533 0.563
## [23] 0.000
```

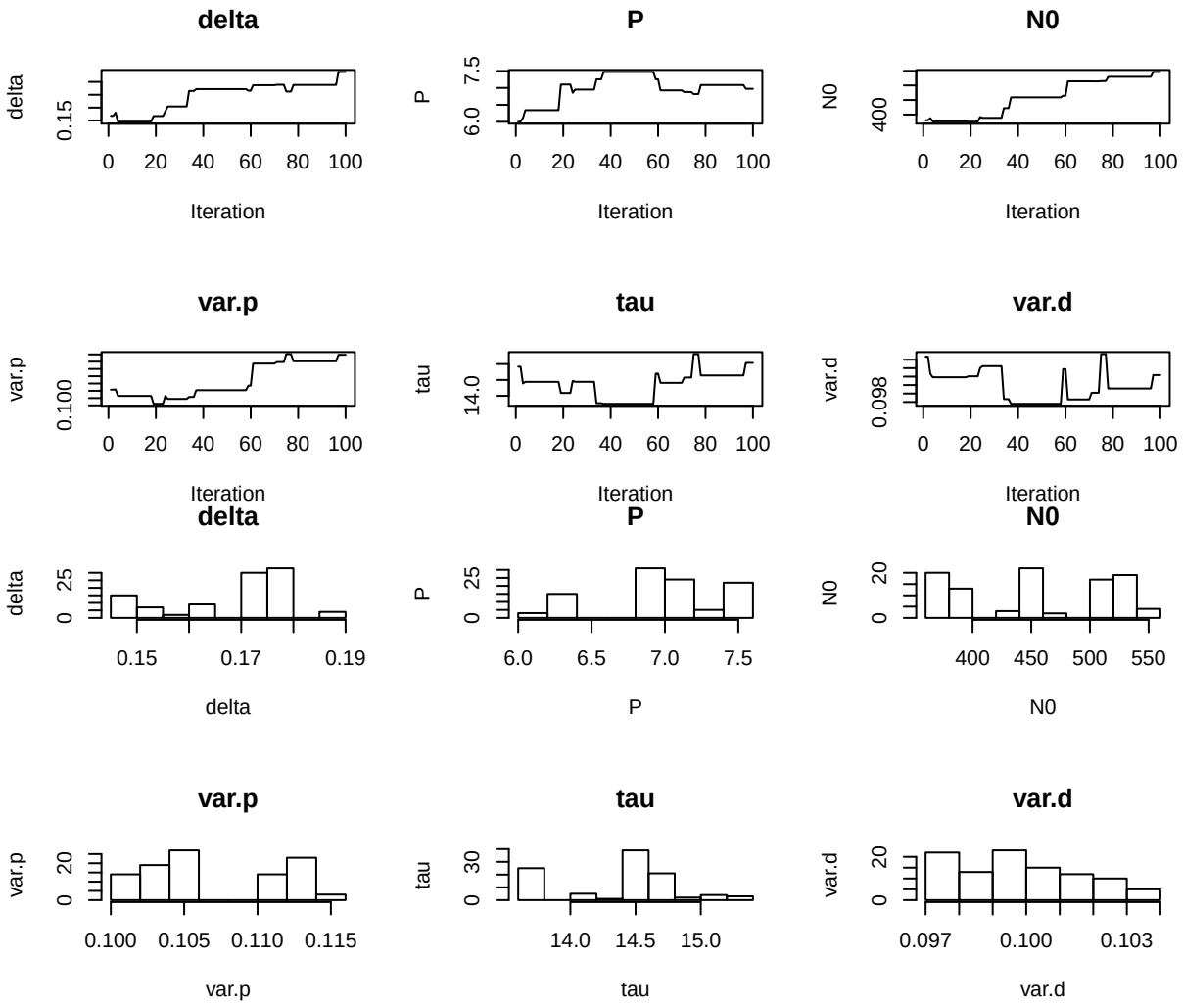
```
slik(blow_sl,
     param = bParams,
     nsim = 1000)
```

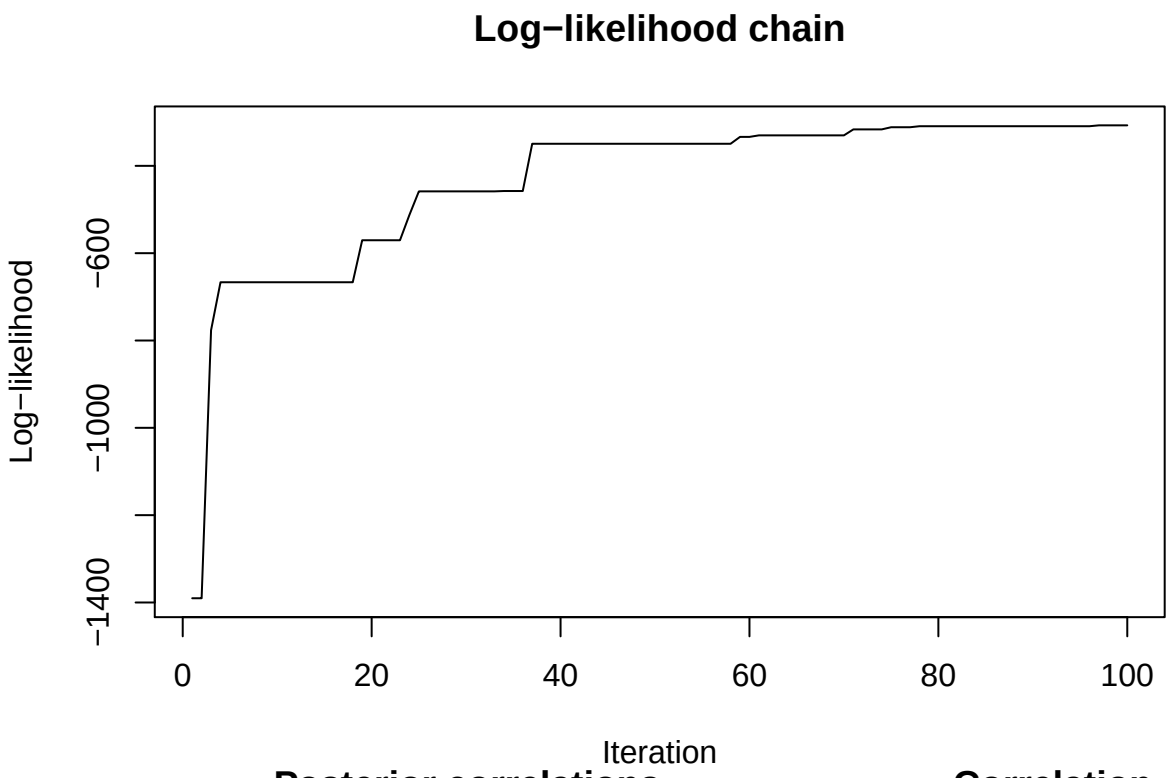
```
## [1] -3297345
```

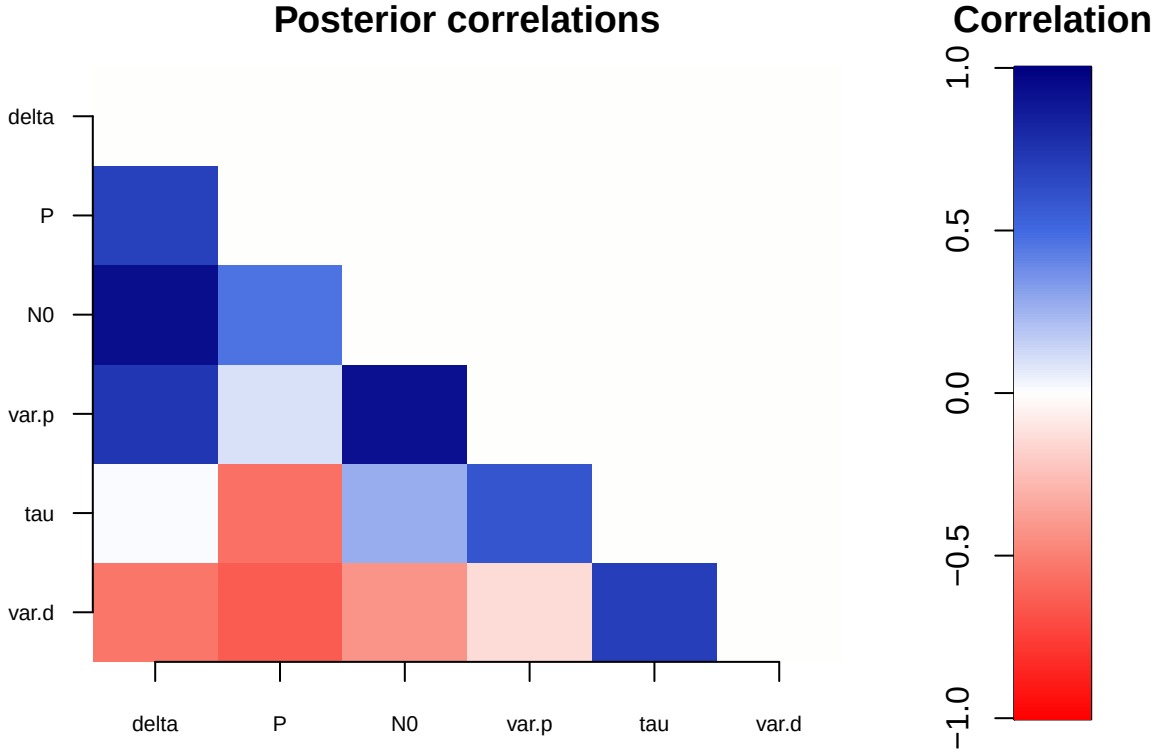
```
blow_sl <- smcmc(blow_sl,
                 initPar = log( bParams ),
                 niter = 100,
                 burn = 0,
                 propCov = diag(rep(0.001, 6)),
                 nsim = 500,
                 prior = function(input, ...){
                     sum(input) +
                         dunif(input[4], log(0.01), log(1), log = TRUE) +
                             dunif(input[6], log(0.01), log(1), log = TRUE)
                 },
                 targetRate = 0.15,
                 multicore = FALSE
                 )
```

```r
#this needs a lot more interations to be useful, but my computer is not up to it
par(mfrow=c(2,2))
readline <- function(...) {} ## no-op
plot(blow_sl, trans = tmpTrans)
```

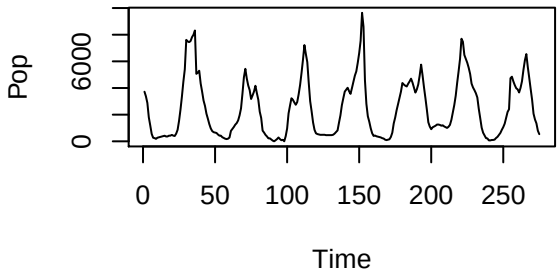
```
## modified .plot.smcmc
```








```
## getMethod("plot",signature=c("synlik","missing"))
## getMethod("plot",signature=c("smcmc","missing"))
## trace("plot",signature=c("synlik","missing"),browser)
```


Pop
0
6000
0
50
100
150
200
250
Time