Exercise 6: Uncertainty and parameter estimation

ETH Zurich Course 701-0426-00L: Modelling Aquatic Ecosystems (Schuwirth)

May 22, 2024

```
# to display chunk code along with its results
knitr::opts_chunk$set(echo = TRUE)
```

Goals:

- Understand the meaning of parameter uncertainty and its propagation to model outputs.
- Conduct error propagation of parameter uncertainty with a simple lake plankton model.
- Apply maximum likelihood parameter estimation.
- Experiment with Bayesian inference of model parameters.

Task 0: Define the simple lake phytoplankton model

```
# load required packages:

if ( !require("ecosim") ) {install.packages("ecosim"); library("ecosim") }
```

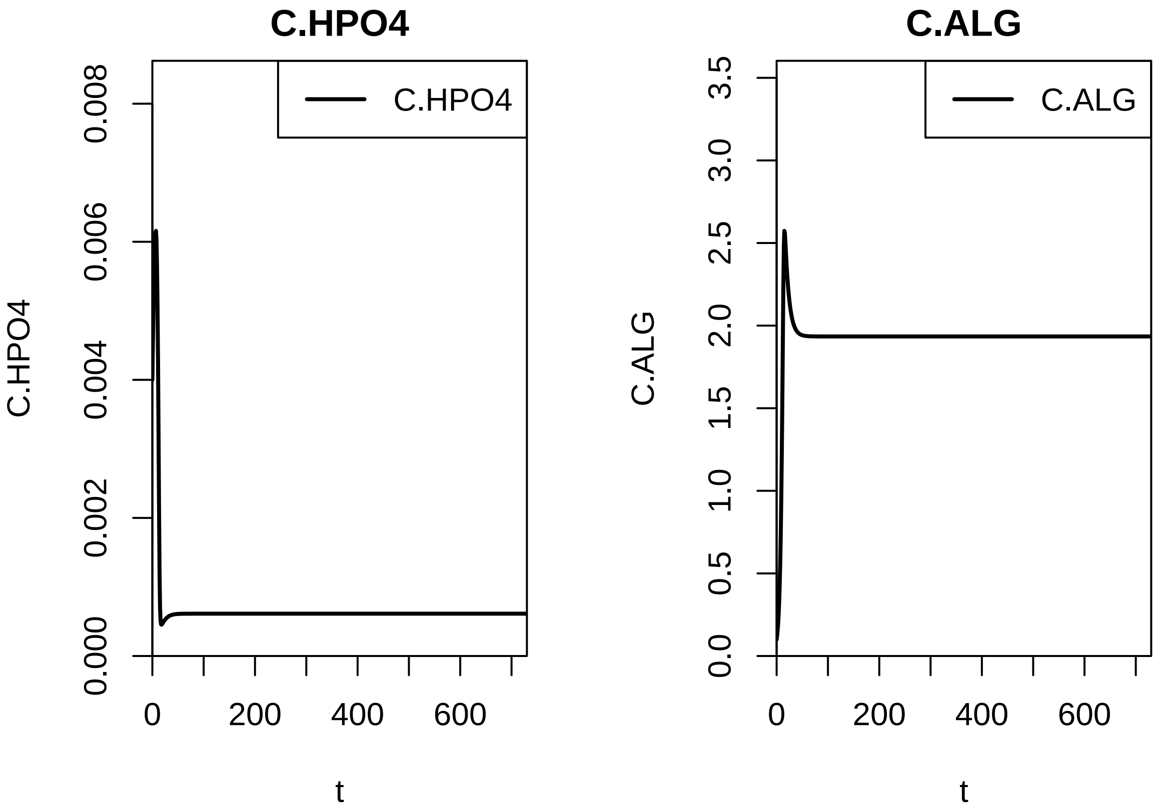
```
## Loading required package: ecosim

## Warning: package 'ecosim' was built under R version 4.3.2

## Loading required package: deSolve

## Warning: package 'deSolve' was built under R version 4.3.2

## Loading required package: stoichcalc
```

```
if ( !require("adaptMCMC") ) {install.packages("adaptMCMC"); library("adaptMCMC") }
```

```
## Loading required package: adaptMCMC

## Warning: package 'adaptMCMC' was built under R version 4.3.3

## Loading required package: parallel

## Loading required package: coda

## Loading required package: Matrix
```

```
# Model with constant driving forces
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# definition of model parameters:

param    <- list(k.gro.ALG   = 0.5,      # 1/d
                 k.death.ALG = 0.1,      # 1/d
                 K.HPO4      = 0.002,    # gP/m3
                 alpha.P.ALG = 0.003,    # gP/gDM
                 A           = 5e+006,   # m2
                 h.epi       = 5,        # m
                 Q.in        = 5,        # m3/s
                 C.HPO4.in   = 0.04,     # gP/m3
                 C.HPO4.ini  = 0.004,    # gP/m3
                 C.ALG.ini   = 0.1)      # gDM/m3

# definition of transformation processes

# growth of algae:

gro.ALG   <- new(Class  = "process",
                 name   = "Growth of algae",
                 rate   = expression(k.gro.ALG
                                     *C.HPO4/(K.HPO4+C.HPO4)
                                     *C.ALG),
                 stoich = list(C.ALG  = expression(1),                # gDM/gDM
                               C.HPO4 = expression(-alpha.P.ALG)))  # gP/gDM

# death of algae:

death.ALG <- new(Class = "process",
                 name   = "Death of algae",
                 rate   = expression(k.death.ALG*C.ALG),
                 stoich = list(C.ALG  = expression(-1)))             # gDM/gDM

# definition of reactor to describe the epilimnion of the lake:

epilimnion <-
   new(Class            = "reactor",
       name             = "Epilimnion",
       volume.ini       = expression(A*h.epi),
       conc.pervol.ini  = list(C.HPO4 = expression(C.HPO4.ini),     # gP/m3
                               C.ALG  = expression(C.ALG.ini)),     # gDM/m3
       inflow           = expression(Q.in*86400),                   # m3/d
       inflow.conc      = list(C.HPO4 = expression(C.HPO4.in),
                               C.ALG  = 0),
       outflow          = expression(Q.in*86400),
       processes        = list(gro.ALG,death.ALG))

# definition of the system consisting of a single reactor:

system <- new(Class    = "system",
              name     = "Lake",
              reactors = list(epilimnion),
```

```
              param    = param,
              t.out    = seq(0,2*365,by=1))

# perform simulation:

res.11.1 <- calcres(system)

# plot results whit default options:

plotres(res.11.1, colnames=list("C.HPO4","C.ALG"))
```

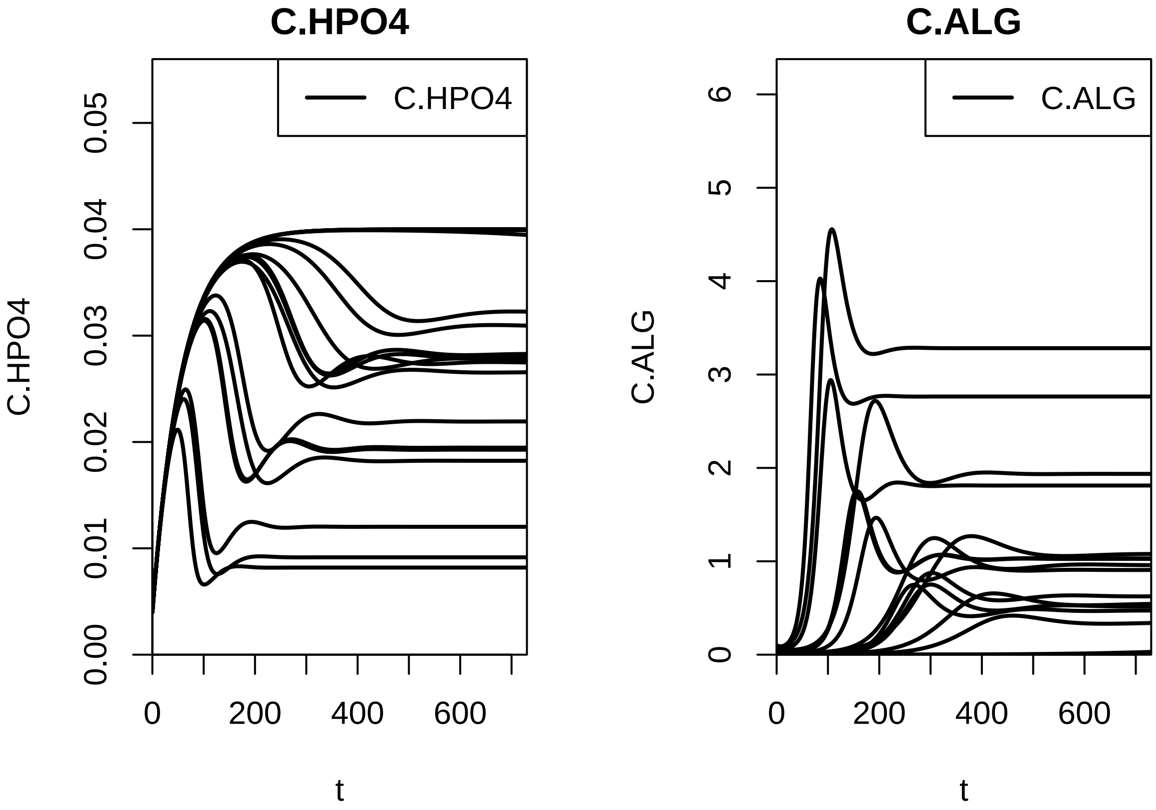


Task 1: Monte Carlo Error propagation of the parameter uncertainty in the simple lake phytoplankton model

Study the implementation of the propagation of the uncertainty of three key parameters of the model.

```
# Task 1:    ###########################################################################################

# Simulation of the simple lake phytoplankton model with consideration of parameter uncertainty
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# set a number of samples we want to draw to forward simulate the uncertainty of the parameters in the
N <- 20  # probably not enough!
```

```r
# initialize a list for the result of the model for each sample forward simulation
res.parunc  <- list()

for ( i in 1:N ){

    # draw parameters from the log-normal distributions
    param.unc <- param
    param.unc[["k.gro.ALG"]] <- randnorm(mean = param[["k.gro.ALG"]],
                                         sd = 0.05, log = TRUE)
    param.unc[["k.death.ALG"]] <- randnorm(mean = param[["k.death.ALG"]],
                                         sd = 0.05, log = TRUE)
    param.unc[["K.HPO4"]] <- randnorm(mean = param[["k.death.ALG"]],
                                         sd = 0.05, log = TRUE)

    # simulate and save model results with new parameter values
    system@param <- param.unc
    res.parunc[[i]] <- calcres(system)

}

# plot results of all simulations with parameters uncertainty together
plotres(res      = res.parunc,
        colnames = list("C.HPO4","C.ALG"))
```



Questions/Tasks:

- How would you decide on the standard deviation for the different parameters?
- How large has N to be to get stable results?
- (optional) Compute the mean and sd of the outputs at t = 365
- (optional) Make a histogram of the model outputs at t = 365

Task 2: Formulation of a likelihood function for the simple lake phytoplankton model

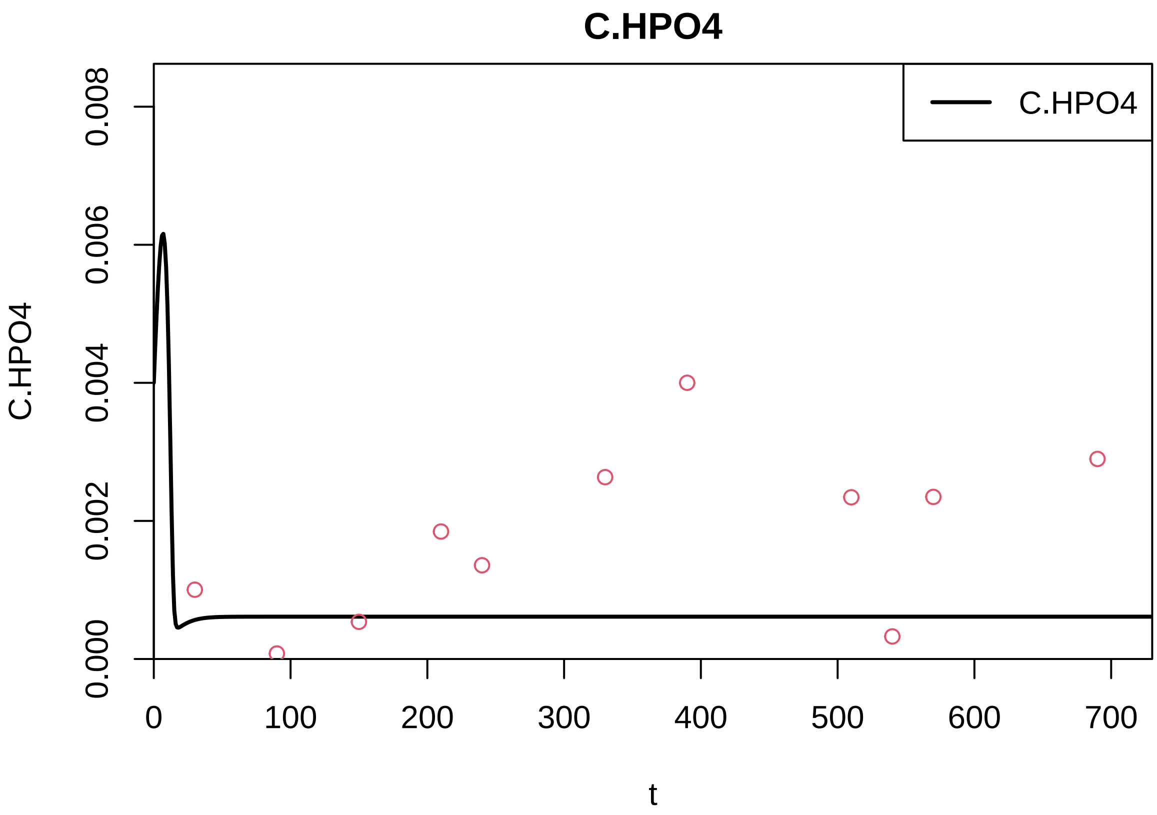
A probabilistic model is formulated as an assumed probability distribution of observations conditional on the values of the model parameters. We exemplify this concept with the simplest possible approach that assumes that the observations are distributed independently and normally, centered at the predictions of the model for the given parameters. Note that the probabilistic models can have additional parameters. In this case it is the standard deviations of the normal distributions. We assume these standard deviations to be different for every output variable, but constant over time.

The probability distribution of the observations given model parameters is a function of both observations and model parameters. When used as a function of the model parameters with actual data substituted for the observations, this function is called “likelihood function’ ’.

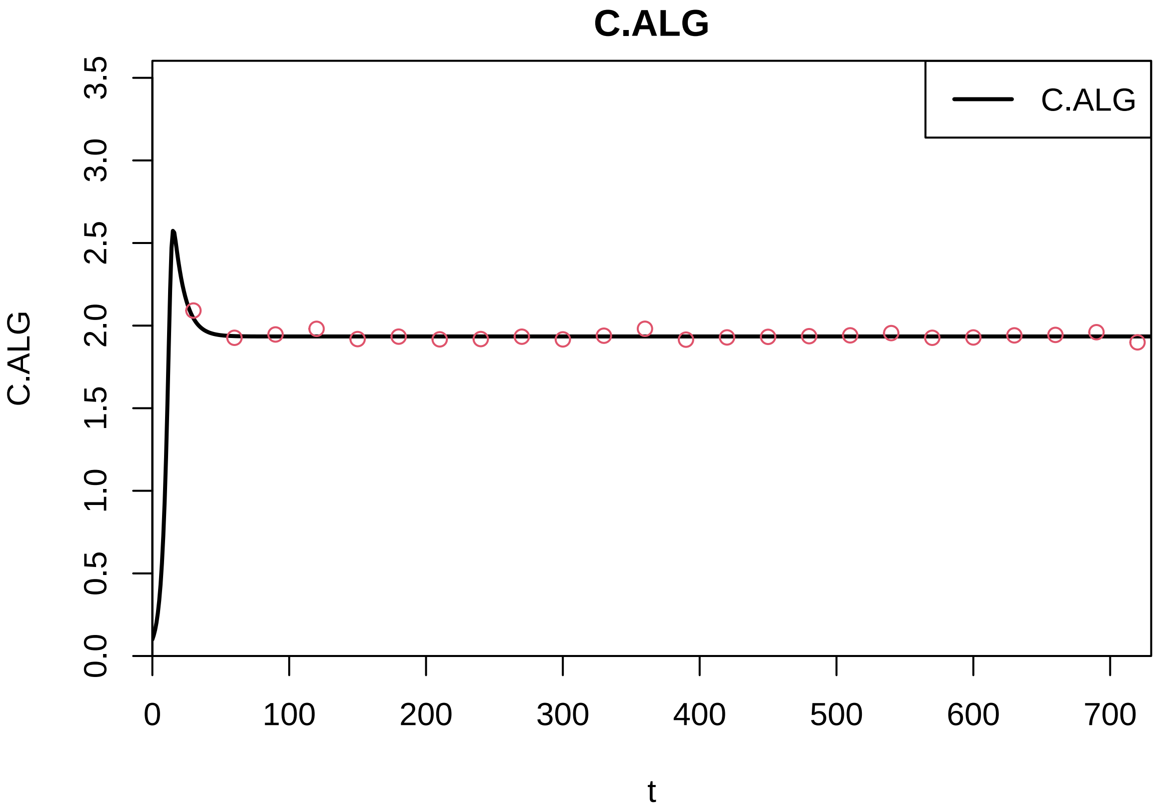
Study the implementation of the likelihood function as outlined above.

First, we load our observations and plot them against model results:

```
# load data
file.name <- "Observations_Concentration_HPO4_ALG.csv"
observations <- read.csv(file.name, header = T, sep = ",", stringsAsFactors = F)
rownames(observations) <- observations$Day # replace row names of the dataframe to match the structure

# plot data

plotres(res      = res.11.1,
        colnames = "C.HPO4")
points(observations$Day, observations$C.HPO4, col=2)
```



```
plotres(res      = res.11.1,
        colnames = "C.ALG")
points(observations$Day, observations$C.ALG, col=2)
```



We now define the likelihood function. Please read it row by row and check whether you understand the comments:

```
# Formulation of a likelihood function for the lake plankton model
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

loglikeli <- function(par, system, obs, verbose=FALSE){
  # negative parameter values lead to a likelihood of zero or a log likelihood of
  # minus infinity:
  if ( any(par<=0) ) return(-Inf)

  # set the parameters equal to the current values given as the first function argument
  # (keep the other parameters):
  system@param[names(par)] <- par

  # set the start time to zero and the other output times to those with observations:
  system@t.out <- c(0,as.numeric(rownames(obs)))

  # calculate the deterministic results of our model:
  res <- calcres(system)

  # calculate the log likelihood using independent, normal distributions
  # with extracting the standard deviations from the parameter vector
  ll <-
    sum(c(dnorm(x=obs[,"C.HPO4"], mean=res[-1,"C.HPO4"], sd=par["sd.obs.HPO4"], log=TRUE),
          dnorm(x=obs[,"C.ALG"] , mean=res[-1,"C.ALG"] , sd=par["sd.obs.ALG"], log=TRUE)))
```

```
  # print parameters and likelihood if the verbose mode was selected:
  if ( verbose ) { print(par); cat("loglikeli =",ll,"\n") }

  # return the log likelihood value
  return(ll)
}
```

Test whether it works:

```
par.ini <- c(k.gro.ALG = 0.5, k.death.ALG = 0.1, K.HPO4 = 0.002,
             sd.obs.HPO4 = 0.004, sd.obs.ALG = 0.02)
loglikeli(par=par.ini, system=system, obs=observations, verbose=TRUE)
```

```
##   k.gro.ALG k.death.ALG      K.HPO4 sd.obs.HPO4  sd.obs.ALG
##       0.500       0.100       0.002       0.004       0.020
## loglikeli = 156.6138
```

```
## [1] 156.6138
```

### Task 3: Maximum likelihood parameter estimation

The aim is to find the parameters that lead to the highest likelihood function. This approach is a very general method of point estimation of parameters. It can be complemented by estimating the uncertainty of the point estimates by confidence intervals or confidence regions. (The theory how to do this is beyond the scope of this course.)

Study the implementation of maximum likelihood parameter estimation from time series of ‘observed'' data using the R functionoptim‘ for numerically maximizing the likelihood function implemented in task 2.

As the likelihood function was already defined in task 2, maximum likelihood parameter estimation only consists of calling a optimizer. Note that this takes some time. Set verbose=TRUE if you want to follow the progress.

```
# Task 3:    ##########################################################################################

# Maximum likelihood parameter estimation
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
res.optim <- optim(par = par.ini, fn = loglikeli,
                   method = "BFGS", control = list(fnscale=-1, maxit=300),
                   system = system, obs = observations, verbose = FALSE)

print("maximum likelihood solution:")
```

```
## [1] "maximum likelihood solution:"
```

```
print(round(res.optim$par, 4))
```

```
##   k.gro.ALG k.death.ALG      K.HPO4 sd.obs.HPO4  sd.obs.ALG
##      0.5000      0.0994      0.0026      0.0042      0.0200
```

```
print("original parameter values:")
```

```
## [1] "original parameter values:"
```

```
print(par.ini)
```

```
##   k.gro.ALG k.death.ALG      K.HPO4 sd.obs.HPO4  sd.obs.ALG 
##       0.500       0.100       0.002       0.004       0.020
```

Questions/Tasks:

- What happens if you choose different initial values `par.ini`?
- How do you interpret `sd.obs.HPO4` and `sd.obs.ALG`?

We proceed with Bayesian inference to get posterior probability distributions that describe our knowledge including its uncertainty.

### Task 4: Bayesian parameter estimation

Bayesian inference combines prior information of parameters with observed data. The result is the updated posterior probability distribution of the parameters.

We first have to define a function that returns the prior probability density for given parameters. For simplicity, we assume independent normal distributions that are trunced as zero (so parameters must be positive):

```
# Task 4:    ###########################################################################################

# Bayesian parameter estimation
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

logprior <- function(par){
  # the prior density is zero for negative values, its log is minus infinity:
  if ( any(par<0) ) return(-Inf)

  # define priors for all parameters that we want to infer
  sum(dnorm(x=par["k.gro.ALG"], mean=0.5,sd=0.1,log=TRUE),
      dnorm(x=par["k.death.ALG"], mean=0.1,sd=0.01,log=TRUE),
      dnorm(x=par["K.HPO4"], mean=0.002,sd=0.0002,log=TRUE),
      dnorm(x=par["sd.obs.HPO4"], mean=0.04,sd=0.004,log=TRUE),
      dnorm(x=par["sd.obs.ALG"], mean=0.02,sd=0.002,log=TRUE)
      )
}

# logposterior = logprior + loglikelihood
logposterior = function(par, obs, system, verbose=FALSE){
    lp = logprior(par)
    if(is.infinite(lp)){
        return(-Inf) # then sampler will go away from these param values (instead of crashing)
    } else {
        lp = lp + loglikeli(par, system=system, obs=obs, verbose=verbose)
```

```
        return(lp)
    }
}
```

Because we are not only interested in the "best" parameters but in the whole posterior distribution, we use a monte Carlo Markov Chain sampler instead of an optimizer. This is constitutionally rather expensive and will take some time. You can reduce the argument `sampsize` to continue and improve the sample later with a longer run.

```
par.ini <- c(k.gro.ALG = 0.5, k.death.ALG = 0.1, K.HPO4 = 0.002,
             sd.obs.HPO4 = 0.004, sd.obs.ALG = 0.02)

# test postior function
logprior(par = par.ini)
```

```
## [1] -17.93368
```

```
logposterior(par = par.ini, system=system, obs=observations)
```

```
## [1] 138.6801
```

```
# run sampler
sampsize <- 1000
res.mcmc <- MCMC(logposterior, n=sampsize,
                 system=system, obs=observations,
                 init = par.ini,  acc.rate = 0.234,
                 scale = 0.00001*par.ini)
```

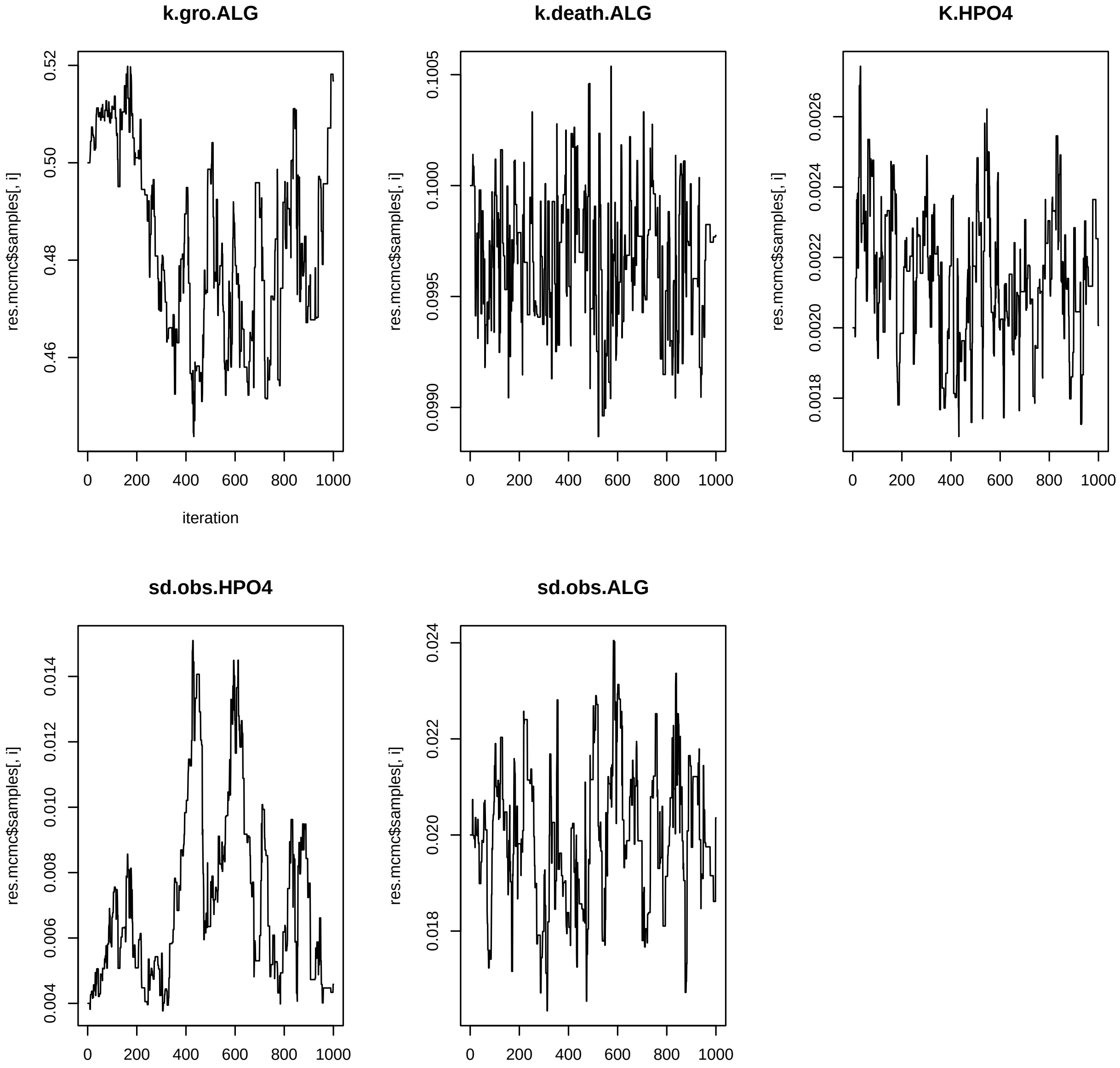
```
##   generate 1000 samples
```

```
res.mcmc$acceptance.rate
```

```
## [1] 0.287
```

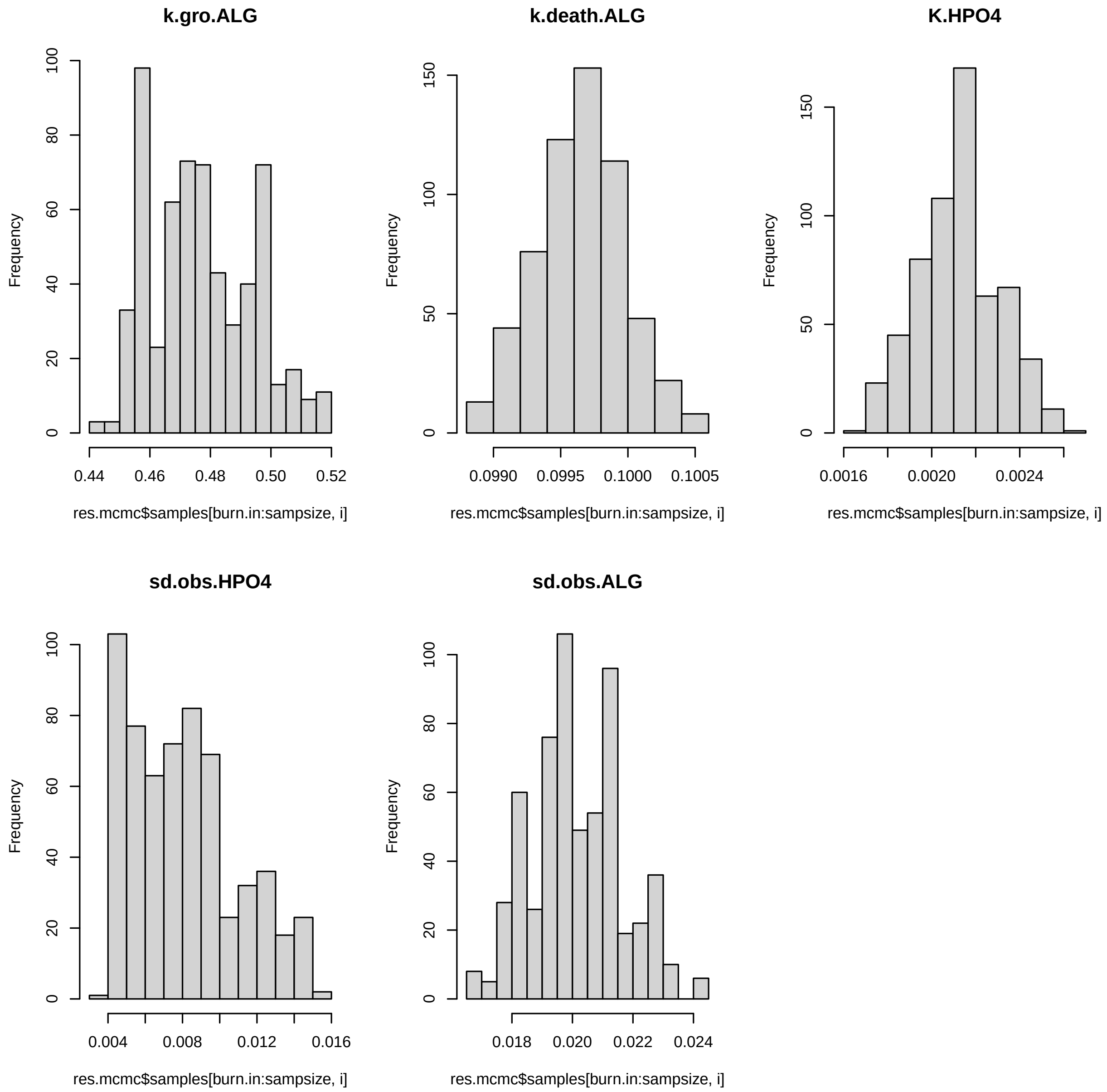
We need to check the chains and decide how much "burn-in" we have to remove:

```
par(mfrow=c(2, 3))
for(i in 1:ncol(res.mcmc$samples)) {
  plot(res.mcmc$samples[,i], type="l", main=colnames(res.mcmc$samples)[i], xlab="iteration"[i])
}
```



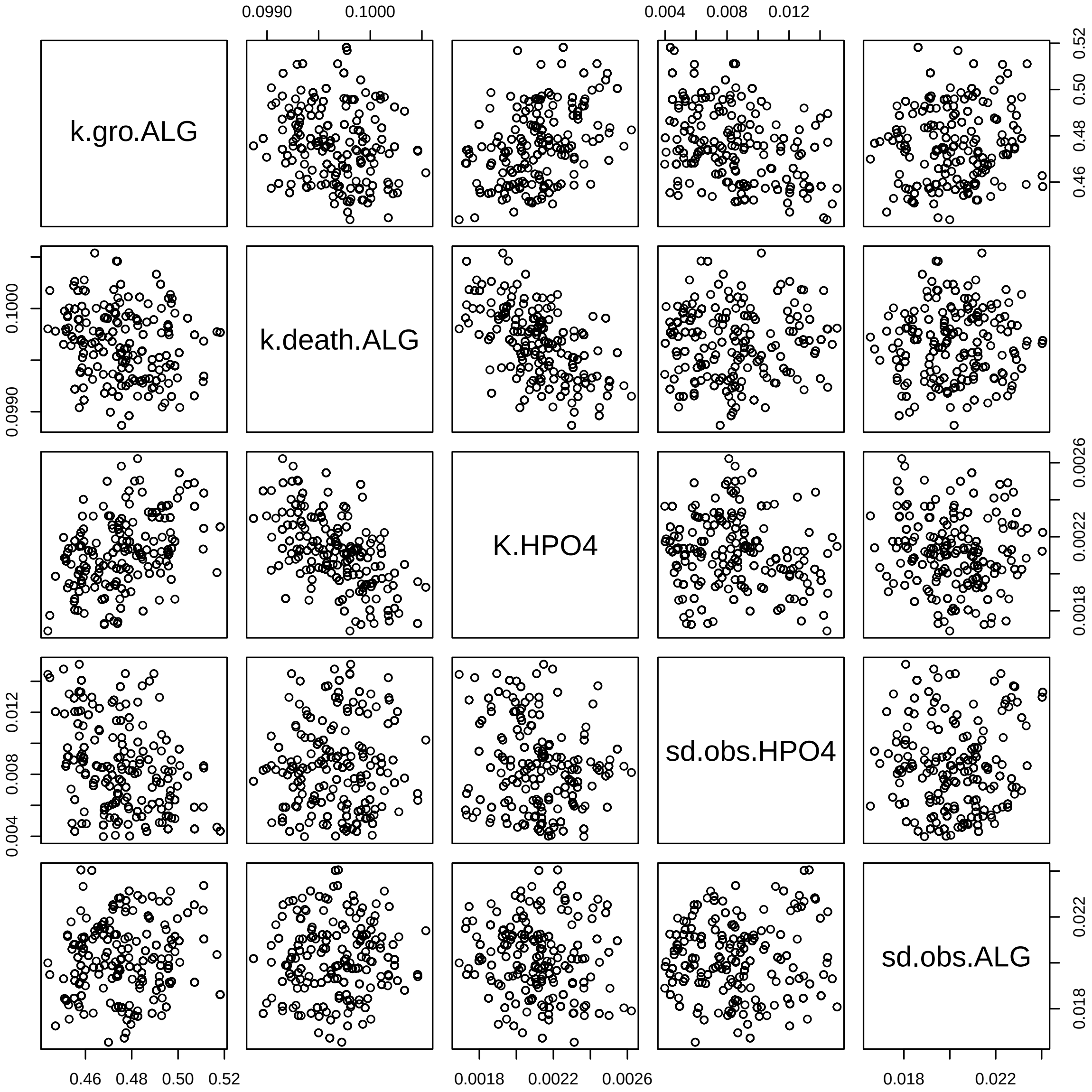
Given the sample, we can plot approximate marginal distributions of all parameters:

```
burn.in <- 400 # remove the first 400 iterations

par(mfrow=c(2, 3))
for(i in 1:ncol(res.mcmc$samples)) {
  hist(res.mcmc$samples[burn.in:sampsize,i], main=colnames(res.mcmc$samples)[i])
}
```



Or look at the sample of two-dimensional marginals:

```
pairs(res.mcmc$samples[burn.in:sampsize,])
```



Questions/Tasks:

- What can you get out of the parameter histograms?
- Experiment with different prior distributions
- Try different initial values. What happens, if the initial values are very far off?
- What happens if the sample size is too small?
- Are some parameter correlated?