MCMC in AD Model Builder

AD Model Builder foundation

anders@nielsensweb.org

What is MCMC and which variant are we using

- Assume we have an unnormalized probability density function $\phi(\theta)$
- MCMC is a collection of methods to simulate a Markov chain $\theta_1, ..., \theta_N$ with an equilibrium distribution given by $\phi(\theta)$
- This is probably known to some from WinBUGS
- AD Model Builder uses what is known as a RW-MH (Random Walk Metropolis-Hastings)
- The starting point is $\hat{\theta}$ and the proposal variance is $\widehat{\text{var}(\hat{\theta})}$

Example: The negative binomial

- Assume that these 15 numbers follow a negative binomial distribution:

```
13 5 28 28 15 4 13 4 10 17 11 13 12 17 3
```

- The model is coded as:

```
DATA_SECTION
  int N
  !! N=15;
  init_vector X(1,N)

PARAMETER_SECTION
  init_number logsize;
  init_bounded_number p(0,1);
  sdreport_number size;
  sdreport_number pp;
  objective_function_value nll;

PROCEDURE_SECTION
  size=exp(logsize);
  pp=p;
  nll=-sum(gammln(X+size))+N*gammln(size)+
       sum(gammln(X+1.0))-N*size*log(p)-sum(X)*log(1.0-p);
```

```
 index    name      value      std dev
     1    logsize   1.3017e+00 4.7101e-01
     2    p         2.2218e-01 8.5571e-02
     3    size      3.6754e+00 1.7312e+00
     4    pp        2.2218e-01 8.5571e-02
```

Basic use

- Simply run the model with `-mcmc N`, where N is the number of steps. For instance:

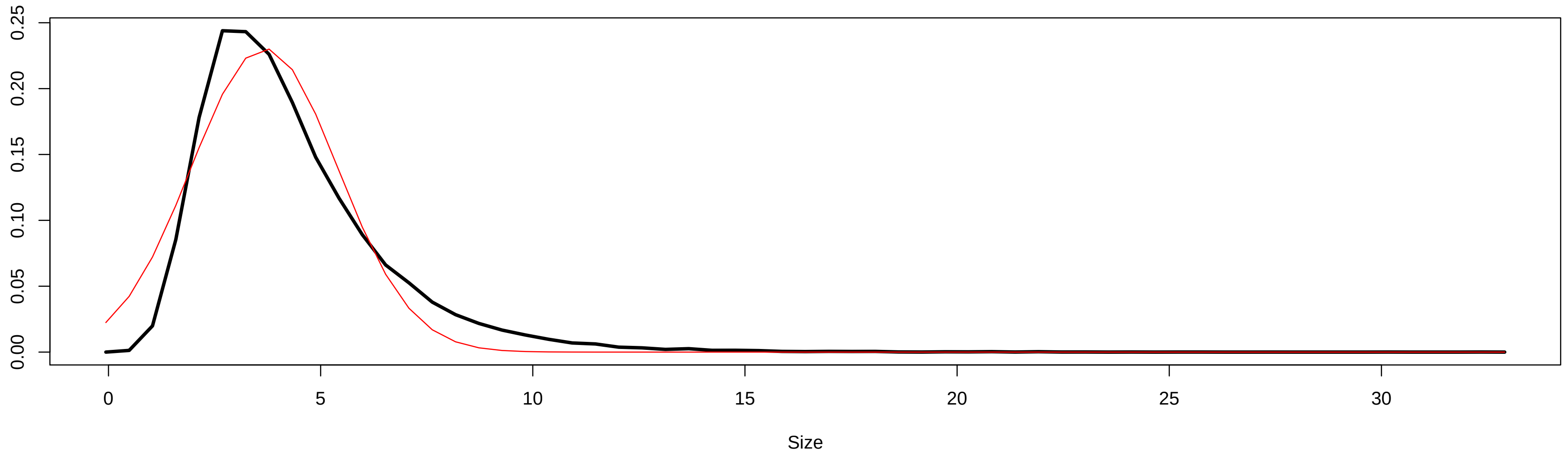
```
an@ch-pcb-an:~$./simplenbin -mcmc 10000
```

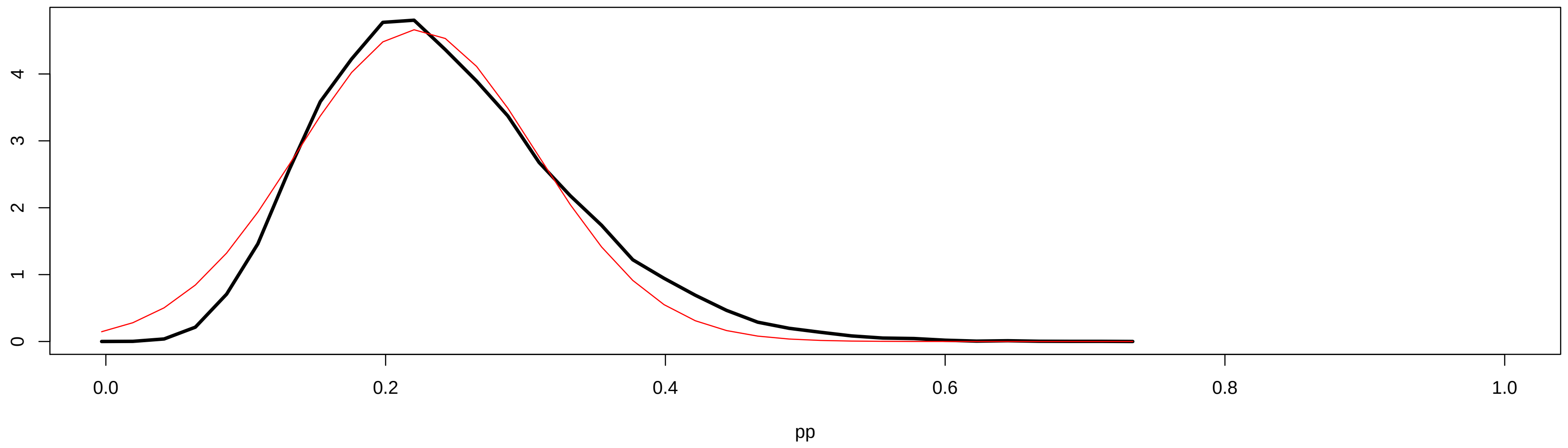
- The file `<modelname>.hst` then contains points on the simulated pdf of all sdreport variables.

```
# samples sizes
10000
# step size scaling factor
1.2
# step sizes
 0.549441 0.0223384
# means
 4.33401 0.242679
# standard devs
 4.39827 0.178819
# lower bounds
 -8 -11
# upper bounds
 34 19
#number of parameters
2
#current parameter values for mcmc restart
 0.826337 0.148048
#random nmber seed
1262173905
```

```
#size
-0.0615169 0
0.487924 0.00127402
1.03736 0.0251165
...
22.4655 0.000182003
23.015 0

#pp
-0.00304368 0
0.0192947 0.00447659
0.0416332 0.0358127
...
0.64477 0.00447659
0.667109 0
```

Useful for plotting





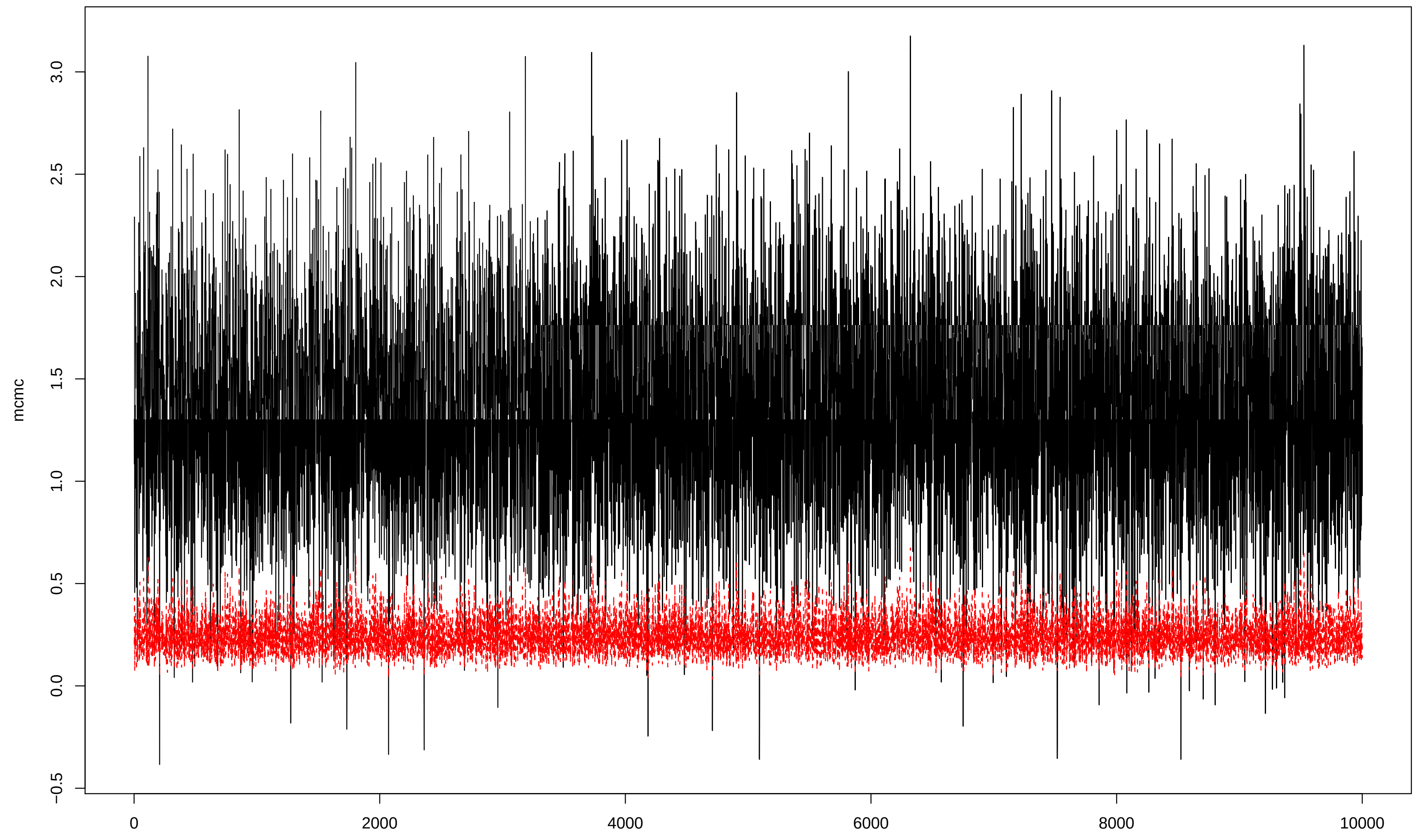
Want to study the chain?

- The chain of parameters (not sdreport variables) can be saved by:

```
an@ch-pcb-an:~$./simplenbin -mcmc 100000 -mcsave 10
```

- here the `-mcsave N` tells it to save every N'th step
- Saves to a binary file `<modelname>.psv`, which can be read into R by:

```
> filen <- file("MCMC/simplenbin.psv", "rb")
> nopar <- readBin(filen, what = integer(), n = 1)
> mcmc <- readBin(filen, what = numeric(), n = nopar * 10000)
> mcmc <- matrix(mcmc, byrow = TRUE, ncol = nopar)
```


mcmc
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
2000
4000
6000
8000
10000

The chain of custom output

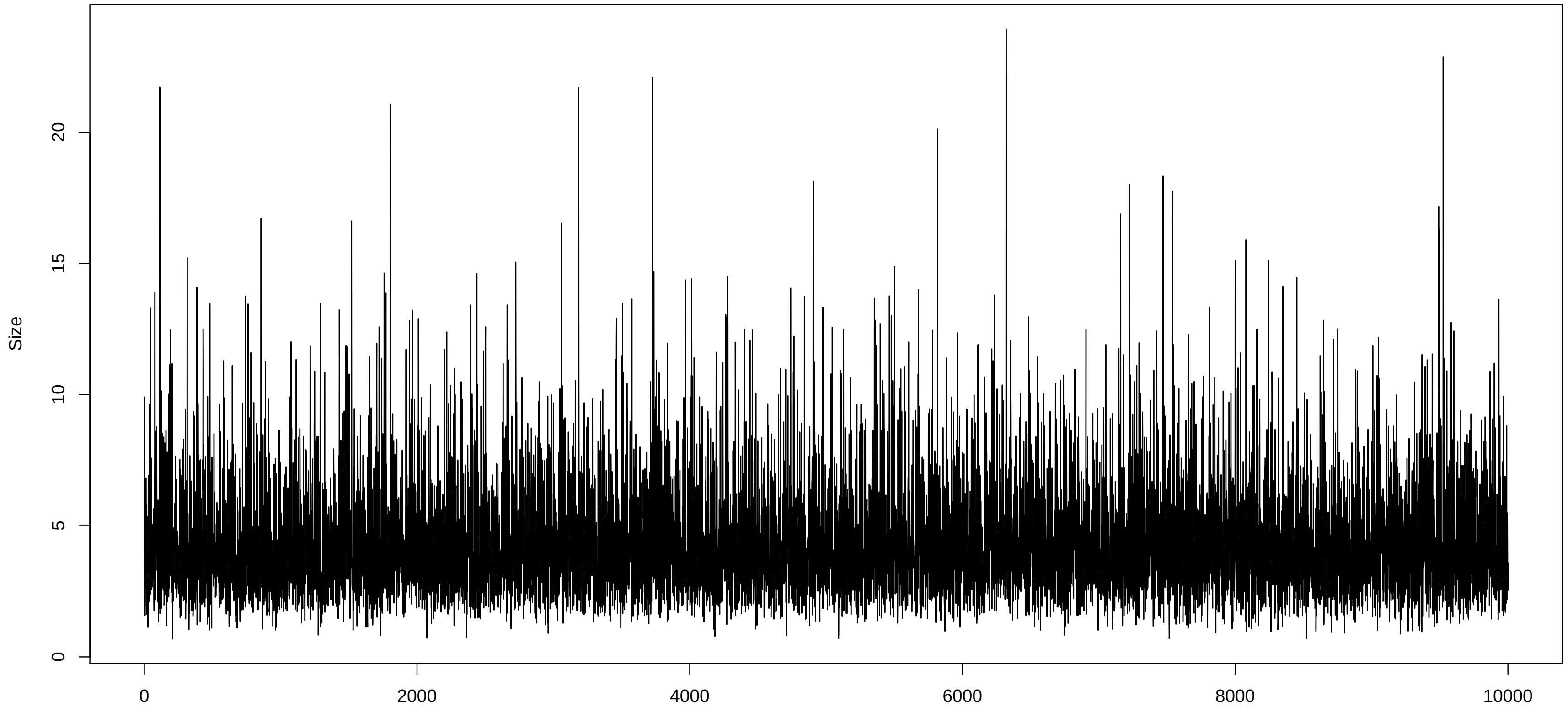
- Suppose we want the output chain of something that is not a model parameter (here 'size')
- Then we need to change the code a bit

```
GLOBALS_SECTION
  #include <fstream.h>
  ofstream sizeout("size.cha");

DATA_SECTION
  int N
  !! N=15;
  init_vector X(1,N)

PARAMETER_SECTION
  init_number logsize;
  init_bounded_number p(0,1);
  sdreport_number size;
  sdreport_number pp;
  objective_function_value nll;

PROCEDURE_SECTION
  size=exp(logsize);
  pp=p;
  nll=-sum(gammln(X+size))+N*gammln(size)+
       sum(gammln(X+1.0))-N*size*log(p)-sum(X)*log(1.0-p);

  if(mceval_phase()){
    ofstream sizeout("size.cha", ios::app);
    sizeout<<size<<"\n";
  }
```

- To run we must type:

```
an@ch-pcb-an:~$./simplenbin -mcmc 100000 -mcsave 10
an@ch-pcb-an:~$./simplenbin -mceval
```

- And then `size.cha` is produced



Exercise: MCMC on Beverton-Holt model

- Try the MCMC on the Beverton-Holt model from yesterday, and plot the joint distribution of $\log(a)$ and $\log(b)$