# Quantifying uncertainties

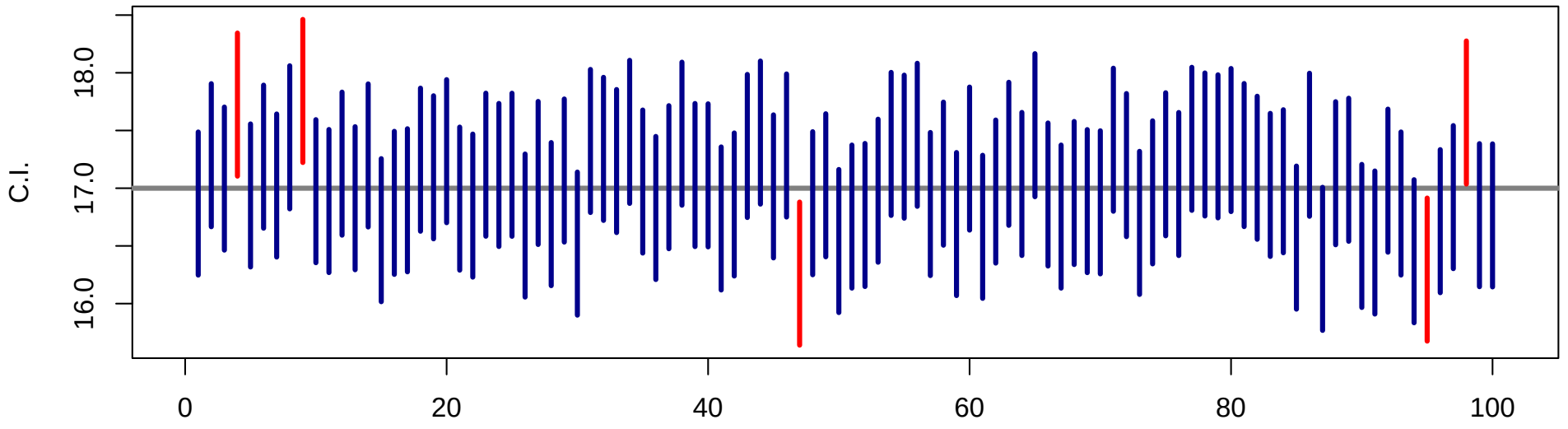
Anders Nielsen & Ben Bolker

an@aqua.dtu.dk

# Let's do an experiment

Consider $x_1, ..., x_{10}$ i.i.d. $N(\mu = 17, \sigma^2 = 1)$ with $\sigma$ known and $\mu$ to be estimated.

```
> set.seed(12345)
> sim <- replicate(100, { x <- rnorm(10,17,1); ci <- mean(x)+c(-1.96,1.96)*sqrt(1/10) } )
```




- This is how we should always think about confidence intervals
- The confidence interval is the random quantity — not the truth!

# Uncertainty comes from the experiment / data collection

- Once we have described the statistical model everything else follows consistently from the model
- Uncertainties in estimated quantities, distributions of test statistics, confidence intervals, ...
- In frequentist statistical inference we always think like:

  *What would happen if we repeated the experiment N times?*
- Theoretically we can always simulate to get an answer (may be impossible in practice)

# Joint example: Distribution of an estimator

- The Beverton-Holt stock-recruitment model can be written (slightly re-parameterized) as:

$$\log R_i = \log(a) + \log(\text{ssb}_i) - \log(1 + \exp(\log(b))\text{ssb}_i) + \varepsilon_i \text{ where } \varepsilon_i \sim \mathcal{N}(0, \sigma^2) \text{ independent}$$

- The code in the file `bh.R` and the observations in the file `bh.dat` can be used to estimate the model parameters.
- Simulate new observations, and thereby simulate the distribution of the parameter estimates
- We need to be very aware of scoping, so we do it together
- Are all of the estimators unbiased?
- What about the untransformed parameters $a$, $b$, and $\sigma$?

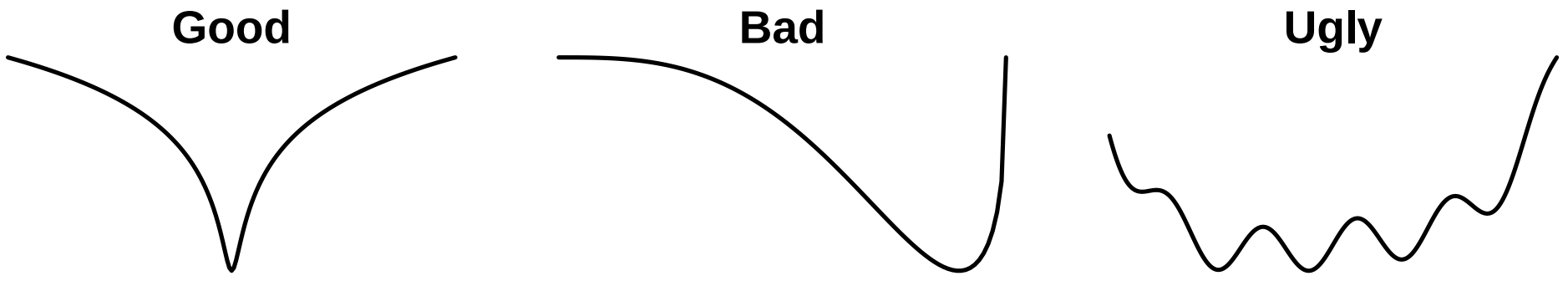
**Note:** A cool feature of RTMB is that you can simulate from the model without any extra real coding! (A) Inform the model about what the observations are e.g: `logR <- OBS(logR)` in the objective function. (B) Use `obj$simulate()`

# Remember the Hessian

- The curvature of the negative log likelihood function gives an asymptotic estimate of the variance of the maximum likelihood estimator:

$$\text{var}(\widehat{\theta}) = \left( \frac{\partial^2 \ell(y|\theta)}{\partial \theta^2} \Big|_{\theta=\widehat{\theta}} \right)^{-1}$$

- The matrix $\mathcal{H}(\widehat{\theta}) = \left( \frac{\partial^2 \ell(y|\theta)}{\partial \theta^2} \Big|_{\theta=\widehat{\theta}} \right)$ is often referred to as "the Hessian matrix"
- Asymptotically we know that $\widehat{\theta} \sim \mathcal{N}(\theta_0, \mathcal{H}(\theta_0))$, but in practice we may be far from the asymptotic behaviour.
- In `RTMB` the `sdreport` gives us the Hessian-based uncertainty estimates



# The delta method for derived quantities

- If we are willing to assume:

$$\hat{\theta} \sim \mathcal{N}(\theta, \Sigma)$$

- and are interested in a quantity $Q$, which is a non-linear function of $\theta$:

$$Q = f(\theta)$$

- Then the linear approximation is:

$$Q \sim \mathcal{N}\left(f(\theta), \nabla f(\theta)^T \Sigma \nabla f(\theta)\right)$$

- In practice we plug in $\hat{\theta}$ and the inverse estimated Hessian for $\theta$ and $\Sigma$ respectively
- In `RTMB` we signal with `ADREPORT` that we want the delta approximation as output in `sdreport`

# Profile likelihood

- The profile likelihood for a given scalar parameter $\theta_i$ is defined as:

$$L_p(\theta_i) = \max_{\theta_{1,\ldots,i-1,i+1,\ldots,n}} L(\theta)$$

  (note that the index $i$ is not in there)

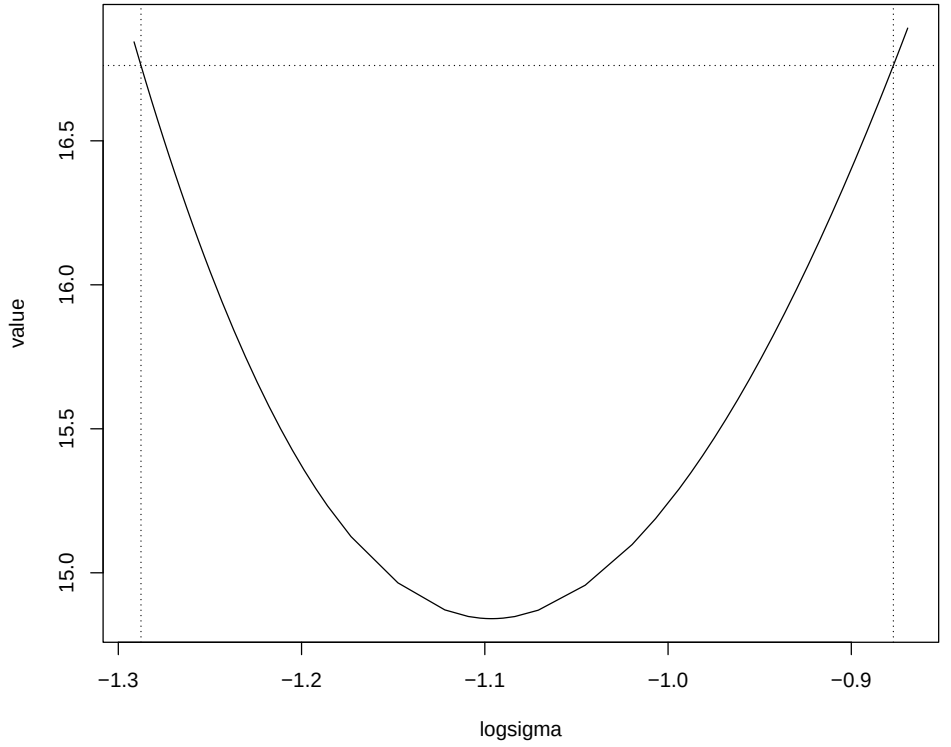
- so while keeping $\theta_i$ at some fixed value we optimize the likelihood w.r.t. all other parameters
- We can then calculate that for a lot of different values of $\theta_i$
- That can show us the shape of the (profile) likelihood w.r.t. that parameter
- We can get a parameterization-invariant confidence interval for $\theta_i$
- Corresponds 1:1 to a hypothesis test for the value of that parameter

# Example: Profile likelihood for $\log \sigma$ in Beverton-Holt

- Profile likelihood for a given model parameter is easily accessible via TMB

```
library(RTMB)
dat <- read.table("bh.dat", header=TRUE)
par <- list(loga=0, logb=0, logsigma=0)

nll <- function(par) {
  getAll(par, dat)
  sigma <- exp(logsigma)
  pred <- loga+log(ssb)-log(1+exp(logb)*ssb)
  nll <- -sum(dnorm(logR,pred,sigma,TRUE))
  a <- exp(loga)
  ADREPORT(a)
  return(nll)
}

obj <- MakeADFun(nll, par, silent=TRUE)
opt <- nlminb(obj$par, obj$fn, obj$gr)

ls.pro <- TMB:::tmbprofile(obj, "logsigma")
plot(ls.pro)
confint(ls.pro)


#
```

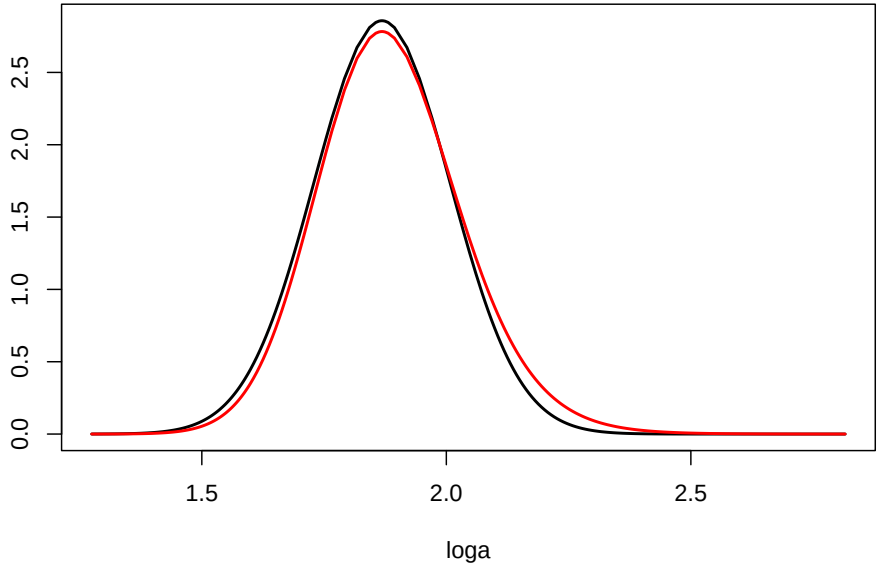
```
               lower      upper
logsigma -1.287628 -0.8770576
```



# Profile likelihood (cont.)

- the points on the y-axis are -logL, so if we want to plot densities we need to plot (`pro[,1], exp(-pro[,2])`)
- If we further need the area under the curve to be $\approx 1$, then we can do:

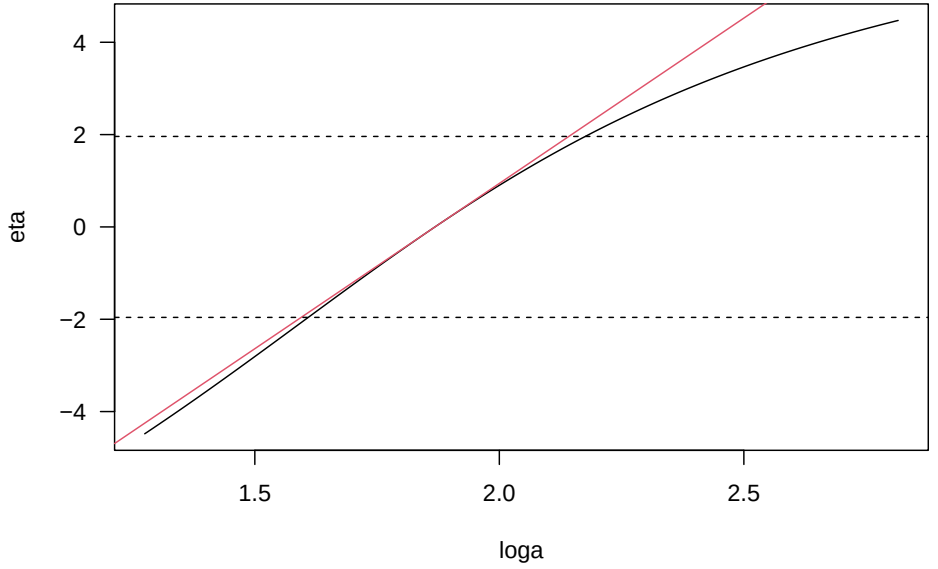
```
density.tmbprofile <- function(x,...){
  xx <- x[,1]
  yy <- exp(-x[,2])
  A <- sum(diff(xx) * (yy[-length(yy)]+yy[-1]))/2
  return(list(x=xx,y=yy/A, data.name=names(x)[1]))
}
pro <- TMB:::tmbprofile(obj,"loga",ytol=10)
sdrep<-summary(sdreport(obj))
plot(pro[,1],dnorm(pro[,1],sdrep["loga",1], sdrep["loga",2]),
     type="l", lwd=2, xlab="loga", ylab="")
lines(density(pro), lwd=2, col="red")
```



# Profile likelihood (cont.)

We can also examine the *signed square root* of the profile, $\mathrm{sgn} \cdot \sqrt{2(\log L_{\max} - \log L(a))}$. This puts the plot on the standard-deviation scale, and makes it easier to distinguish a non-quadratic log-likelihood surface.

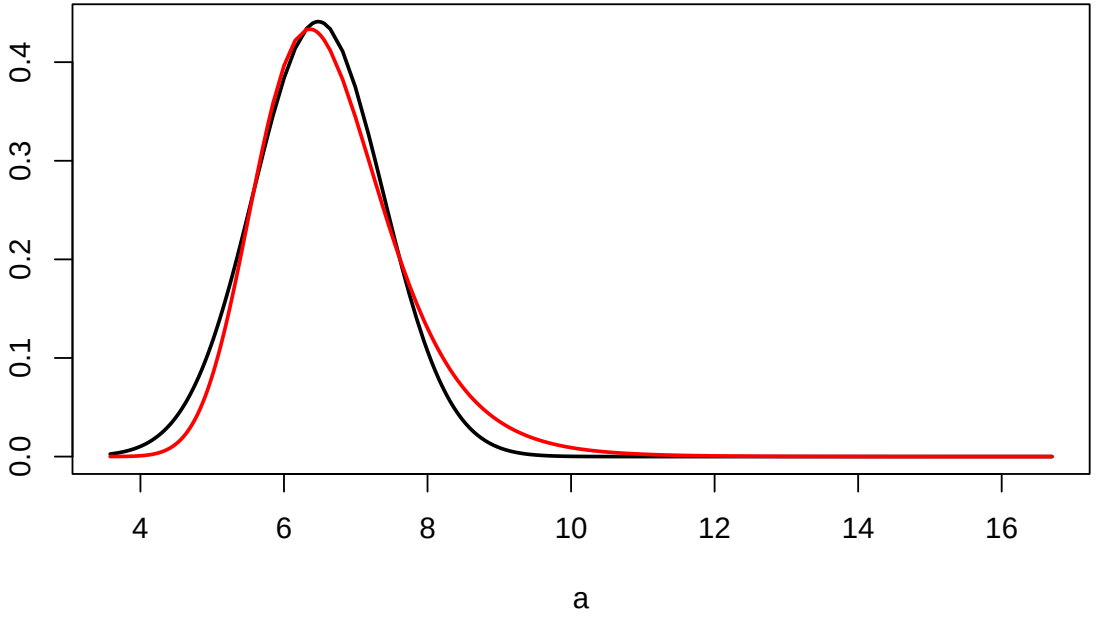
```
loga_est <- sdrep["loga", 1]
loga_sd <- sdrep["loga", 2]
pro_sqrt <- within(pro, {
    vmin <- min(value)
    sgn <- 2*as.numeric(loga>loga_est)-1
    eta <- sgn*sqrt(2*(value-vmin))
})
plot(eta ~ loga, data = pro_sqrt, type = "l", las = 1)
abline(a = -loga_est/loga_sd, b = 1/loga_sd, col = 2)
abline(h = c(-1,1)*1.96, lty = 2)
```



# Profile likelihood (cont.)

- How about for $a$ itself?

```
pro <- TMB:::tmbprofile(obj,"loga",ytol=10)
d<-density(pro)
pro[,1]<-exp(d$x)
pro[,2]<-d$y/exp(d$x)
plot(pro[,1],dnorm(pro[,1],sdrep["a",1], sdrep["a",2]),
     type="l", lwd=2, xlab="a", ylab="")
lines(pro, lwd=2, col="red")
```



# What is MCMC, and which variant are we using?

https://chi-feng.github.io/mcmc-demo/

- 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, JAGS, NIMBLE, or Stan
- TMB can use the MCMC algorithms from Stan, including:

  **HMC** the Hamiltonian sampler (see Neal 2011)

  **NUTS** the No-U-Turn sampler (see Hoffman and Gelman 2014)

- We will briefly look at an example

# Example: The negative binomial ex via MCMC

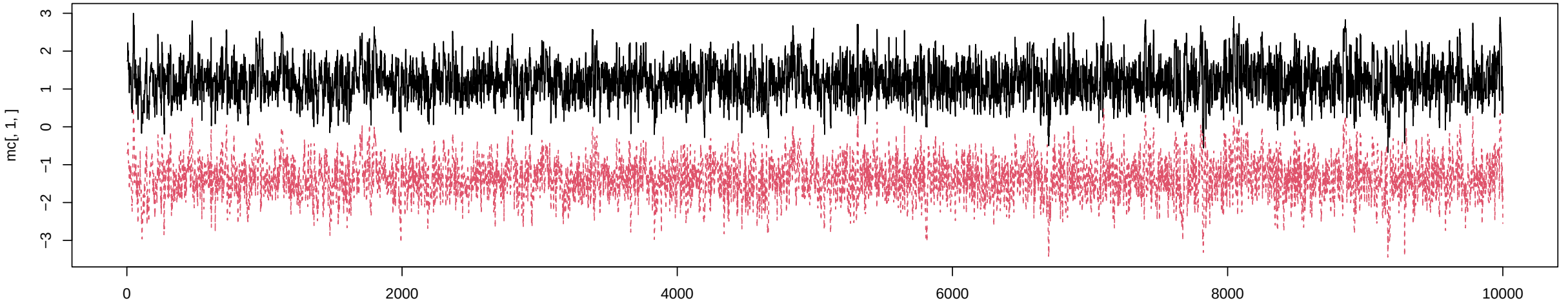
- Assume that these 15 numbers follow a negative binomial distribution:
  13, 5, 28, 28, 15, 4, 13, 4, 10, 17, 11, 13, 12, 17, 3

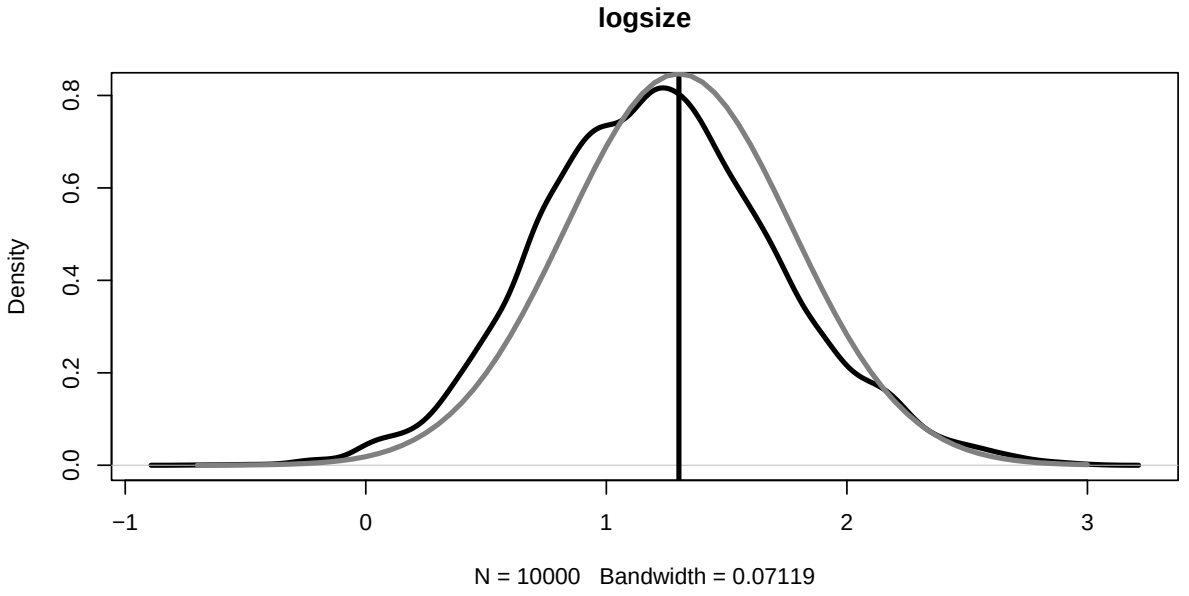
```
library(RTMB)
dat <- list()
dat$Y <- c(13, 5, 28, 28, 15, 4, 13, 4, 10, 17, 11, 13, 12, 17, 3)

par <- list()
par$logsize <- 0
par$logitp <- 0

nll<-function(par){
  getAll(par)
  size <- exp(logsize)
  p <- plogis(logitp)
  -sum(dnbinom(dat$Y, size=size, prob=p, log=TRUE))
}

obj <- RTMB::MakeADFun(nll, par)
opt <- nlminb(obj$par, obj$fn, obj$gr)
sdr <- summary(sdreport(obj))

library(tmbstan)
fitmcmc2 <- tmbstan(obj, chains=1,
              iter=10000, init=list(opt$par),
              lower=c(-50,-50), upper=c(50,50) )
```

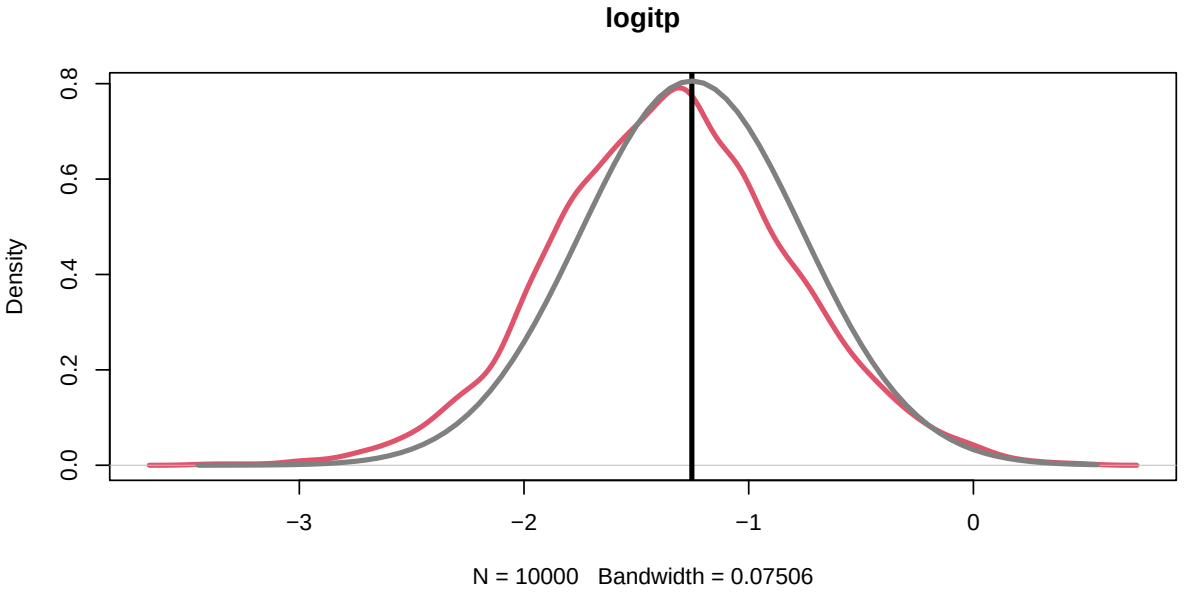
files/mcmc1.R

```
mc <- extract(fitmcmc2, pars=names(obj$par),
              inc_warmup=TRUE, permuted=FALSE)

npar<-dim(mc)[3]
layout(rbind(rep(1,npar),c(2:(npar+1))))
matplot(mc[,1,], type="l")

for(i in 1:npar){
  pn <- attr(mc,"dimnames")$parameters[i]
  plot(density(mc[,1,i]), main=pn, col=i, lwd=3)
  abline(v=sdr[i,1], lwd=3)
  xx <- pretty(range(mc[,1,i]),100)
  lines(xx,dnorm(xx,sdr[i,1],sdr[i,2]),
        col=gray(.5), lwd=3)
}
```

files/mcmc1.R


mc[, 1, ]
3
2
1
0
−1
−2
−3
0
2000
4000
6000
8000
10000


logsize
Density
0.0
0.2
0.4
0.6
0.8
−1
0
1
2
3
N = 10000 Bandwidth = 0.07119


logitp
Density
0.0
0.2
0.4
0.6
0.8
−3
−2
−1
0
N = 10000 Bandwidth = 0.07506

# Same example manually (Random walk Metropolis–Hastings)

- Assume we want to sample from a distribution with density proportional to $\psi(\theta)$
- $\theta \in ]-\infty;\infty[^m$ can be multidimensional, but assumed unbounded
- If we start with one $\theta_0$, then we can suggest another by adding multivariate normal noise
- Then we can accept or reject the suggested value based on the relative likelihood
- The algorithm goes:
  1. Start with an initial parameter $\theta_0$ (and set $i = 1$)
  2. Propose the next by $\theta_i^\star \sim \mathcal{N}(\theta_{i-1}, \Sigma)$
  3. Calculate relative likelihood $\alpha = \frac{\psi(\theta_i^\star)}{\psi(\theta_{i-1})}$
  4. Accept the proposal $\theta_i = \theta_i^\star$ with probability $\min(1, \alpha)$ and reject otherwise $\theta_i = \theta_{i-1}$
  5. Set $i = i + 1$ and and repeat from 2.
- Let's see it in practice

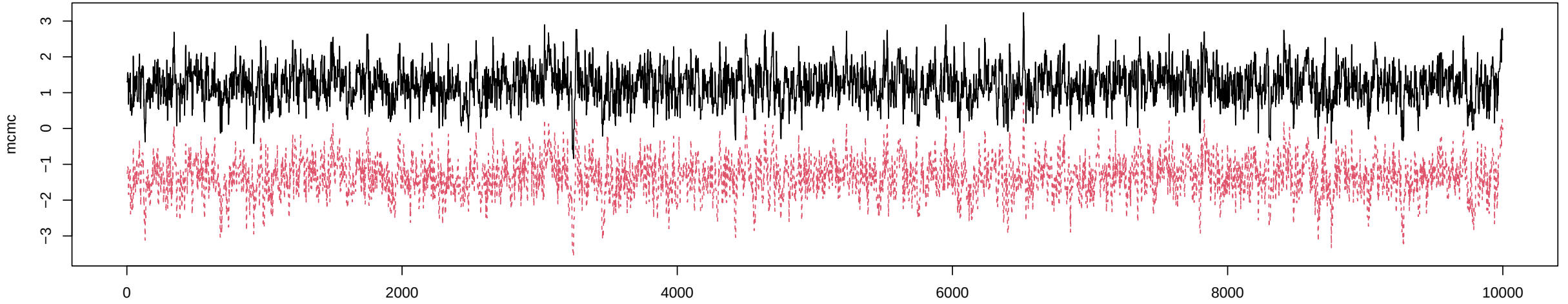
# Example: The negative binomial example manually

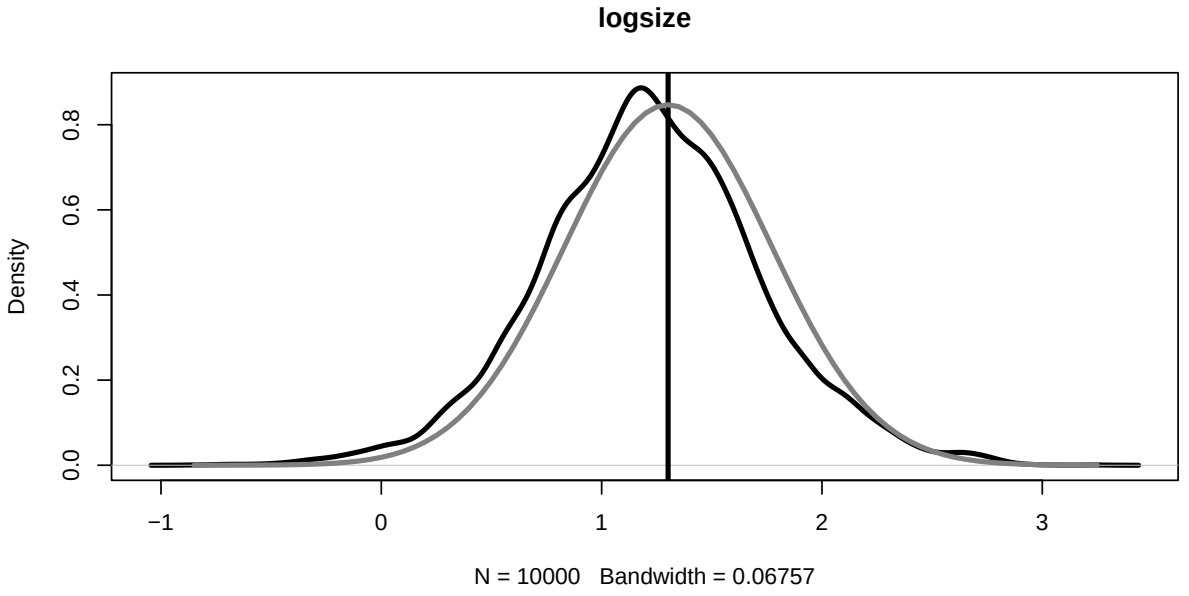
```r
library(RTMB)
dat <- list(Y=c(13,5,28,28,15,4,13,4,10,17,11,13,12,17,3))
par <- list(logsize=0, logitp=0)

nll<-function(par){
  getAll(par)
  size <- exp(logsize)
  p <- plogis(logitp)
  -sum(dnbinom(dat$Y, size=size, prob=p, log=TRUE))
}

obj <- RTMB::MakeADFun(nll, par)
opt <- nlminb(obj$par, obj$fn, obj$gr)
rep <- sdreport(obj)
sdr <- summary(rep)

est <- opt$par
npar <- length(est)
cov <- rep$cov.fixed # or just # diag(npar) #

nosim <- 10000
mcmc <- matrix(NA,nrow=nosim, ncol=npar)
mcmc[1,] <- est
colnames(mcmc) <- names(est)
```

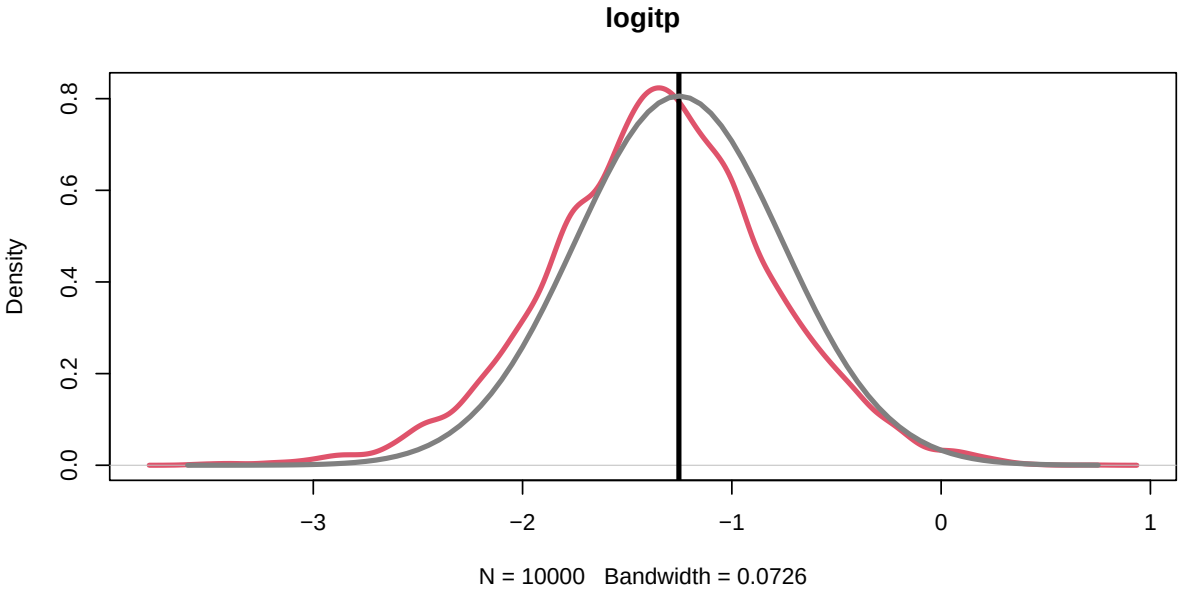
files/mcmc2.R

```r
steps <- MASS::mvrnorm(nosim, mu=rep(0,npar), Sigma=cov)
U <- runif(nosim)

currentValue <- obj$fn(mcmc[1,])
for(i in 2:nosim){
  proposal <- mcmc[i-1,]+steps[i,]
  proposalValue <- obj$fn(proposal)
  if(U[i]<exp(currentValue-proposalValue)){
    mcmc[i,] <- proposal
    currentValue <- proposalValue
  }else{
    mcmc[i,] <- mcmc[i-1,]
  }
}

layout(rbind(rep(1,npar),c(2:(npar+1))))
matplot(mcmc, type="l")
for(i in 1:npar){
  pn <- names(est)[i]
  plot(density(mcmc[,i]), main=pn, col=i, lwd=3)
  abline(v=sdr[i,1], lwd=3)
  xx <- pretty(range(mcmc[,i]),100)
  lines(xx,dnorm(xx, sdr[i,1],sdr[i,2]), col=gray(.5), lwd=3)
}
```

files/mcmc2.R


mcmc
3
2
1
0
−1
−2
−3
0
2000
4000
6000
8000
10000


logsize
Density
0.0
0.2
0.4
0.6
0.8
−1
0
1
2
3
N = 10000 Bandwidth = 0.06757


logitp
Density
0.0
0.2
0.4
0.6
0.8
−3
−2
−1
0
1
N = 10000 Bandwidth = 0.0726

# Exercise: Do MCMC for the Beverton-Holt example

- Use the example in the file `bh.R` and the data `bh.dat` and do MCMC sampling of the 3-dim parameter vector.
- It is up to you if you want do do it via `tmbstan` or manually

# Beverton-Holt: sequential Bayesian vs. integrated analysis

The results shown in the Beverton-Holt example were based only on the data file `bh.dat`

**(a) Sequential Bayesian**

For a similar stock in a similar area we have these estimates

```
> priorMeanLogAB <- c(1.8085,-12.183)
> priorCovLogAB <- rbind(c(0.01619256, 0.03828887),
+                        c(0.03828887, 0.10517698))
```

- Modify the program `bh.R` to use this as prior information.
- Hint: The multivariate normal is available via the function `dmvnorm`

**(b) Integrated analysis**

In the file `bh0.dat` we have the entire data set from a similar stock.

- Modify the program `bh.R` to use both data sets simultaneously to estimate $\log(a)$ and $\log(b)$.

**(c) Compare** Use MCMC and plot the joint distribution of $\log(a)$ and $\log(b)$. Make one plot for the sequential Bayesian approach, and one for the integrated.

# Beverton-Holt comparing sequential Bayesian & integrated analysis