Quantifying uncertainties

Anders Nielsen, Ethan Lawler, & Sean Anderson

an@aqua.dtu.dk

Important to supply uncertainty estimates?

- I can think of three reasons people give for not to presenting uncertainty estimates:
  1. Estimates are so precise that we don't need to worry about uncertainties
  2. Estimates are so uncertain that we prefer not to show uncertainties
  3. We don't know how to estimate the uncertainties, only the estimates themselves.
- The uncertainty estimates are sort of needed in the first case too, and in the last two cases the estimates themselves should be skeptically evaluated.

Confidence interval

The parameter is an unknown constant and no probability statement concerning its value may be made. — Jerzy Neyman, original developer of confidence intervals.

- When dealing with *frequentist* statistical inference — which we do — we always think of probability in terms of repeated experiments.
- The logic is not:
  the true parameter is in this fixed interval with probability 95%
- The logic is:
  An interval calculated like this will cover the true parameter 95% of the times the experiment is repeated

Confidence interval

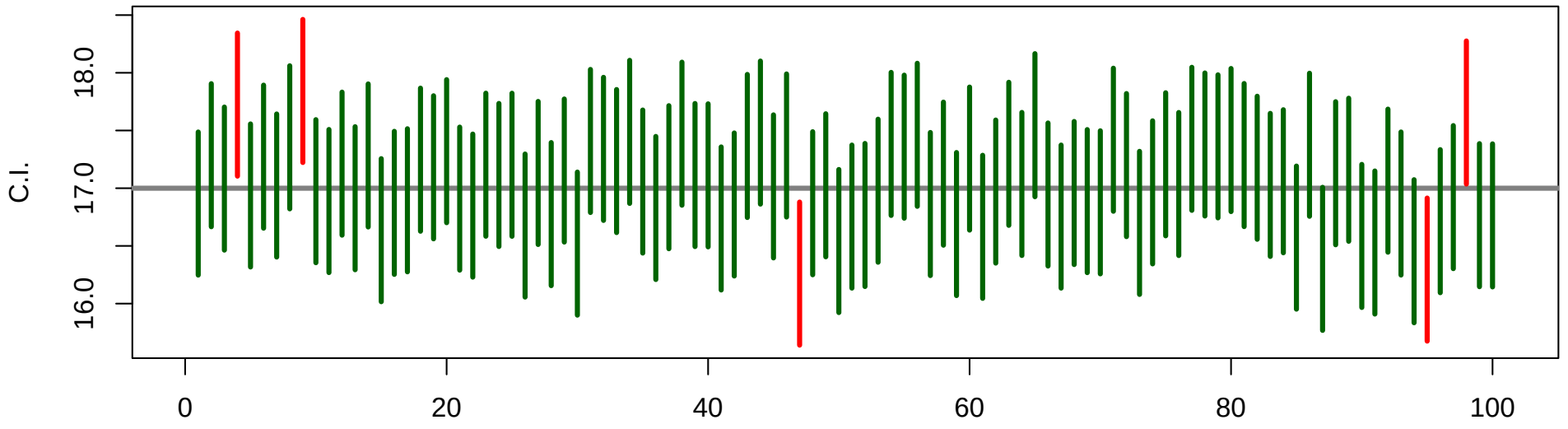
The parameter is an unknown constant and no probability statement concerning its value may be made. — Jerzy Neyman, original developer of confidence intervals.

- When dealing with *frequentist* statistical inference — which we do — we always think of probability in terms of repeated experiments.
- The logic is not:
  ~~*the true parameter is in this fixed interval with probability 95%*~~
- The logic is:
  An interval calculated like this will cover the true parameter 95% of the times the experiment is repeated

Let's do an experiment

Consider $x_1, ..., x_{10}$ i.i.d. $N(\mu = 17, \sigma^2 = 1)$ with σ kown and μ 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)
- Let's try it for the confidence interval of SSB in the assessment model

Exercise: Distribution of an estimator

- The Beverton-Holt model can be written (slightly re-parametrized) 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
- Are all of the estimators unbiased?
- What about the untransformed parameters a, b, and σ?

Hint: 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()`

Bootstrap: new fake simulations

- Simulating from the model is somtimes called 'parametric bootstrap', but what is real bootstrap?
- From out data set $\mathbf{y} = (y_1, \ldots, y_n)$ we make a "new" data set by resampling with replacement
- Real observations

```
> y
 [1] 6 4 4 6 2 6 2 3 3 6 1 3 3 1 1 3 5 4 3 6 3 5 2 6 2 5 1 5 6 5 5 1 4 1 1 5 6 1
[39] 6 6 3 4
```

- A bootstrap sample

```
> idx<-sample(length(y), replace=TRUE) # random sample of index
> y.bootstrap<-y[idx] # a single bootstrap sample
> y.bootstrap
 [1] 4 3 1 5 6 1 3 5 3 1 2 3 6 2 1 1 3 6 1 1 3 3 1 6 2 5 3 5 3 4 5 5 3 3 6 2 5 6
[39] 3 5 1 5
```

- Here we could have sampled directly from y, but the index approach will prove useful later.

Bootstrap method

Based on a single data set $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ make B data sets of the same size by random sampling with replacement.

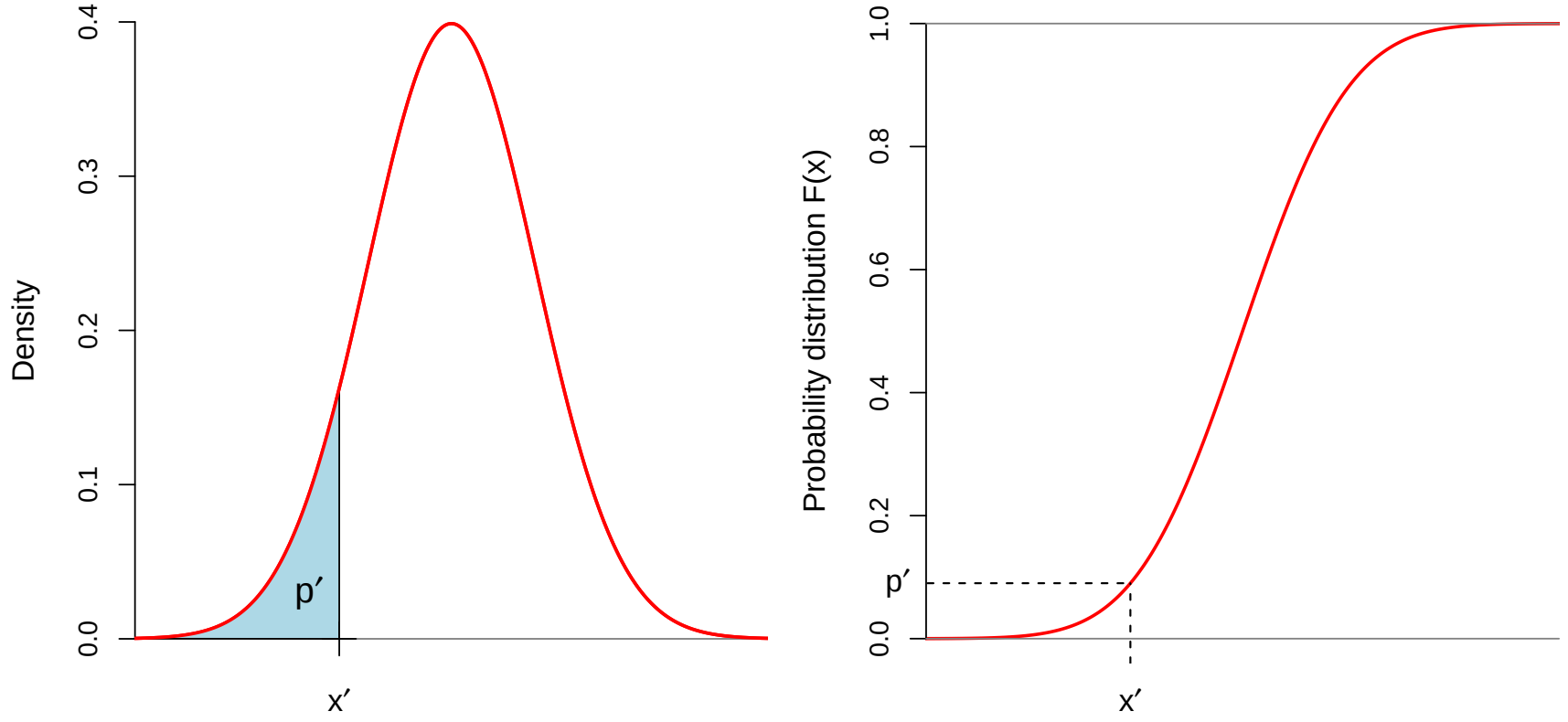
$$\begin{aligned}
\text{BS data set} \quad 1: \quad & (y_1, y_2, \ldots, y_8) \rightarrow \hat{m}_1^* \\
\text{BS data set} \quad 2: \quad & (y_4, y_n, \ldots, y_4) \rightarrow \hat{m}_2^* \\
& \ldots \\
\text{BS data set} \quad B: \quad & (y_9, y_2, \ldots, y_{19}) \rightarrow \hat{m}_B^*
\end{aligned}$$

The B bootstrap estimates can now be used as samples from the distribution of the estimator. From this we can compute e.g. the sd of the estimator.

Why does bootstrap work?

- Assume we have a data set $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ from a known distribution whit cdf $F(y)$.
- We are interested in a derived quantity (e.g. the median)
- As we know the distribution we can simulate B new data sets
- From each data set we can calculate the quantity
- Then we can calculate e.g. sd and quantiles of the quantity

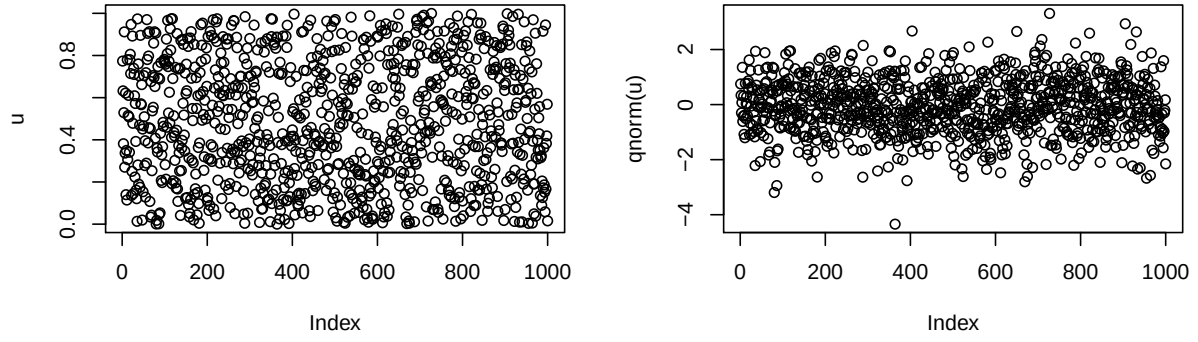
Simulation from a known distribution

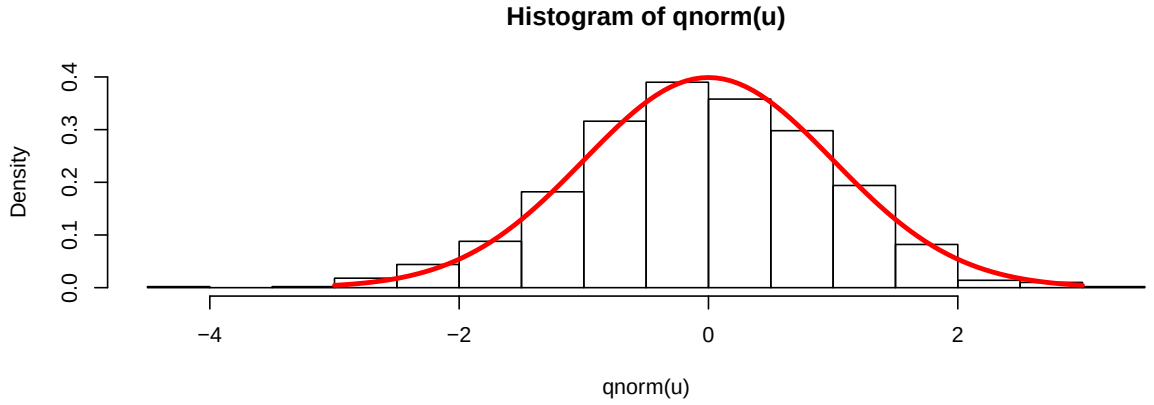


If we want to simulate observations from a distribution with a known distribution function (cdf) F:

1. Simulate n numbers $u_1, \ldots, u_n$ from a uniform distribution between 0 and 1
2. Calculate: $F^{-1}(u_1), \ldots, F^{-1}(u_n)$

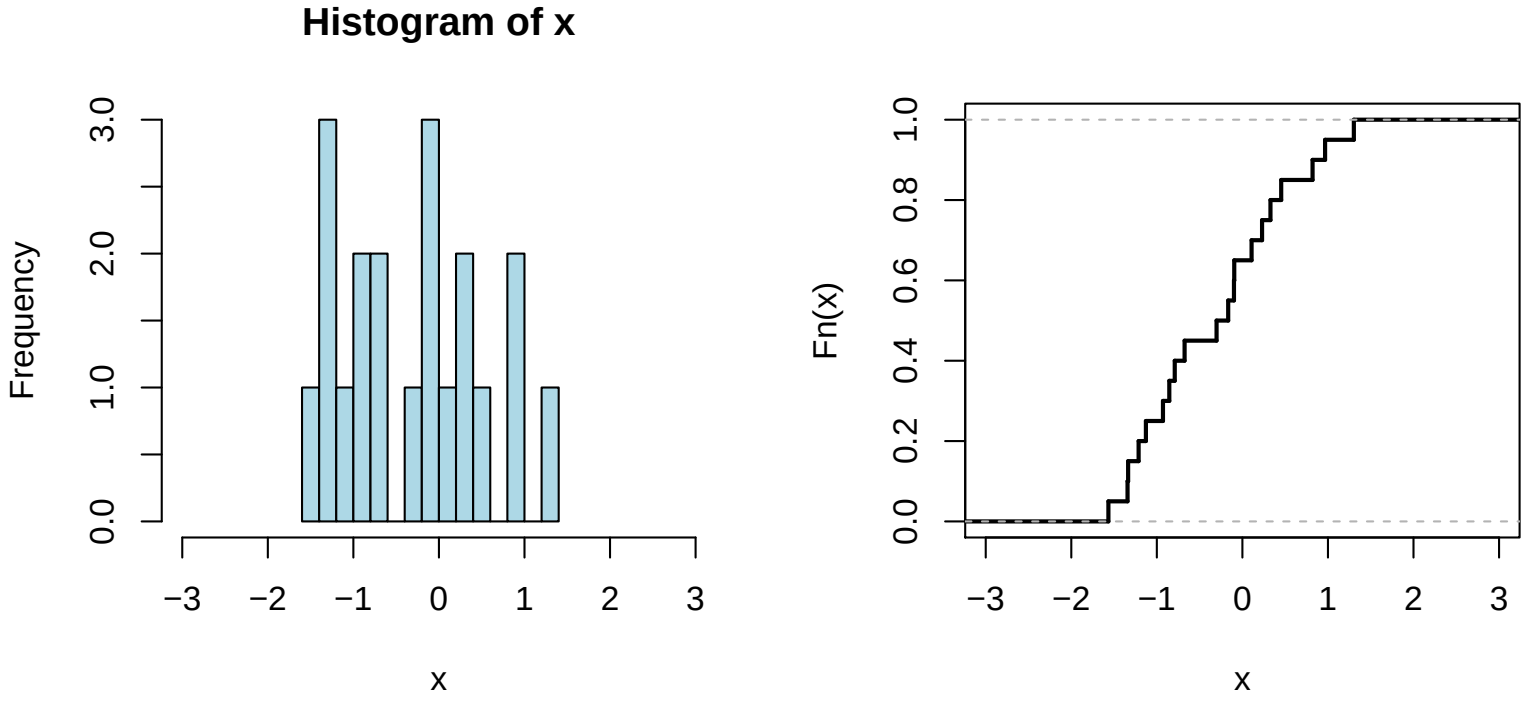
Example: Simulation of a normally distributed sample






```
u<-runif(1000)
plot(u)
plot(qnorm(u))
hist(qnorm(u), prob=TRUE)
curve(dnorm,-3,3, add=TRUE, col='red',lwd=3)
```

The empirical distribution function $\hat{F}$

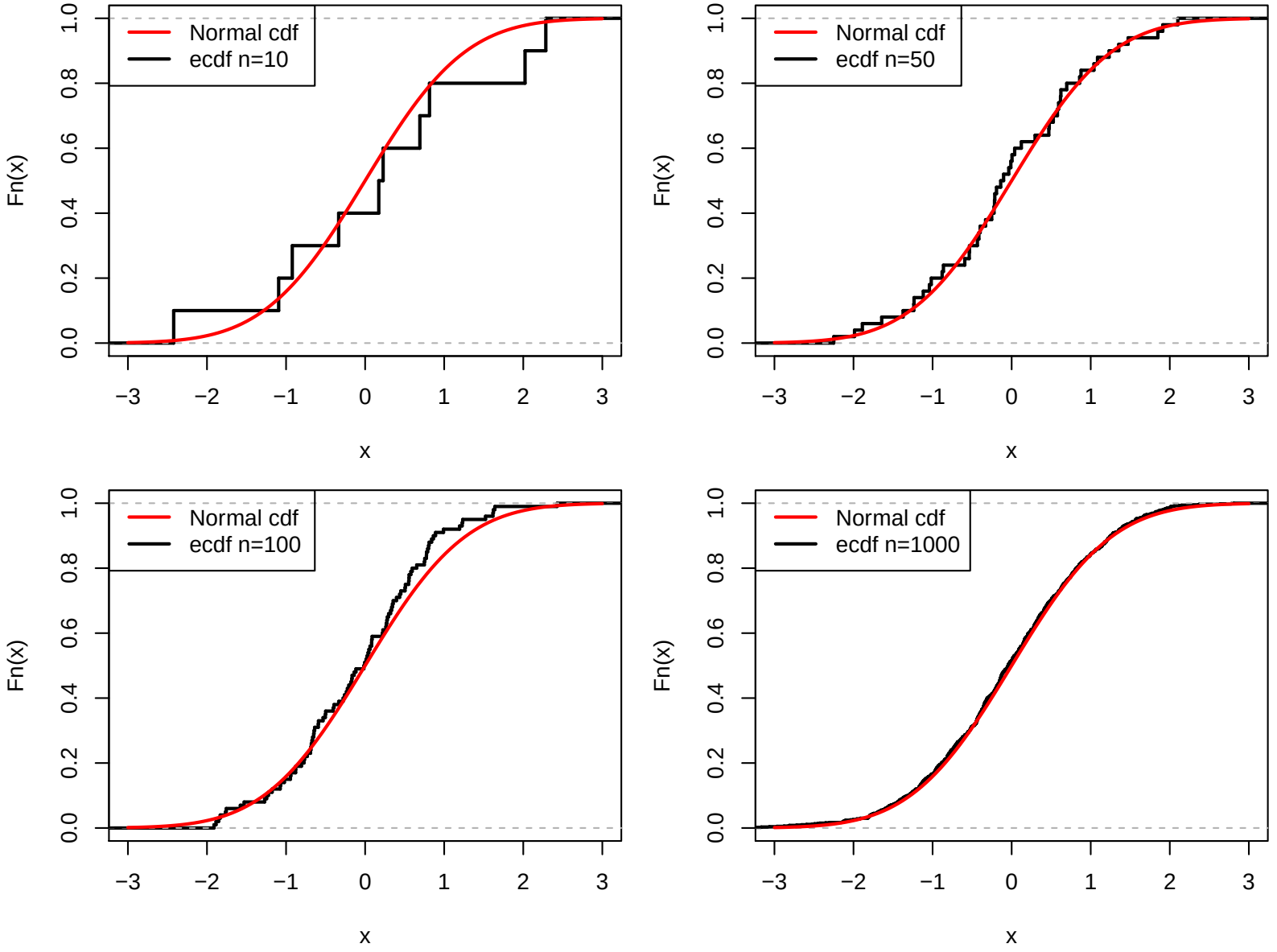


- The logical estimate of the of the distribution function is:

$$\hat{F}(\tilde{x}) = \frac{\#\{x \leq \tilde{x}\}}{n}$$

- Which is simply that our estimate of the probability of e.g. $x \leq 0$ is the fraction of observations below 0

Now consider this:



Exercise: Distribution of an estimator

- Use again the Beverton-Holt example:

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

- where the code in the file `bh.R` and the observations in the file `bh.dat` can be used to estimate the model parameters.
- Compare the distribution of the model parameters between a parametric bootstrap and a real bootstrap based on resampling observations
- The resamples can be obtained by sampling rows of the original data set with replacement
- What are pros and cons of the two approaches?

Joint exercise: Bootstrap in assessment model

- Now consider the basic assessment model
- Can we use bootstrap to explore the uncertainty of SSB
- We can try to re-sample with replacement each observation
- Alternatively we can re-sample observation noise (residuals) within each fleet
- Let's try the approaches together

A confidence interval is a test

- The 95% confidence interval should include the values of the parameters that we would not reject in a test with a 5% significance level.

Likelihood ratio test - general case

- Assume model B is a sub model of model A (this is for instance the case if a free model parameter in A is set to a fixed value in B)
- We can calculate the test statistic $G_{A\to B}$ for reducing model A to model B by:

$$G_{A\to B} = 2(\ell_B(y|\widehat{\theta_B}) - \ell_A(y|\widehat{\theta_A}))$$

- If the two optimal fits are "almost equal" the model reduction is accepted, if the fits are very different the model reduction is rejected
- Asymptotically G followers a χ^2–distribution, so the P-value is given by:

$$P_{A\to B} = P\left(\chi^2_{\mathrm{dim}(A)-\mathrm{dim}(B)} \geq G_{A\to B}\right)$$

- If this is small (often defined as $< 5\%$) the actual observations matches B poorly and the model reduction is rejected.

Maximum likelihood estimator and Hessian — in general

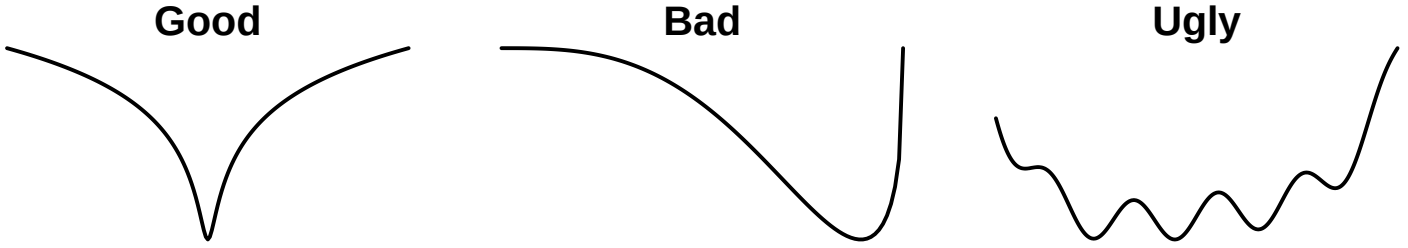
- A sensible estimate of the model parameters is to choose the values that maximize the likelihood for the actual observations.

$$\widehat{\theta} = \underset{\theta}{\operatorname{argmin}}\, \ell(y|\theta)$$

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

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

- 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.



The delta method

- Let's be willing to assume:

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

- Are interested in a quantity, which is a non-linear function of θ:

$$Q = f(\theta)$$

- The linear approximation is:

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

- Mini Exercise: Assume we have estimated $(\log F_2, \log F_3, \log F_4)$ to (-1.13, -0.75, -0.94) with a covariance matrix of:

$$\begin{pmatrix} 0.0222 & 0.0135 & 0.0114 \\ 0.0135 & 0.0169 & 0.0137 \\ 0.0114 & 0.0137 & 0.0191 \end{pmatrix}$$

setup a confidence interval for $\log(\overline{F}_{2-4})$

Hessian based uncertainties for the assessment model

- For any given scalar parameter θ_i we need to pick the corresponding diagonal element in the inverse hessian

$$\widehat{\text{var}(\widehat{\theta}_i)} = (\mathcal{H}(\widehat{\theta})^{-1})_{i,i}$$

- Then we can compute an approximate confidence interval by:

$$\widehat{\theta}_i \pm 2\sqrt{\widehat{\text{var}(\widehat{\theta}_i)}}$$

- Why?
- If it is not a parameter, but a function of one or more parameters, then the covariance matrix in the delta approximation is used instead.
- In practice it is much simpler (demonstrate in `basicfsa.R`)
- Mini exercise: Calculate Hessian based CI in the Beverton-Holt example.

Profile likelihood

- The profile likelihood for a given scalar parameter θ_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 θ_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 θ_i
- That can show us the shape of the (profile) likelihood w.r.t. that parameter
- We can get a parametrization-invariant confidence interval for θ_i

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

- Plot the negative log profile likelihood for $\log \sigma$ in the Beverton-Holt example
- Calculate it 'manually' or via the TMB provided helper function

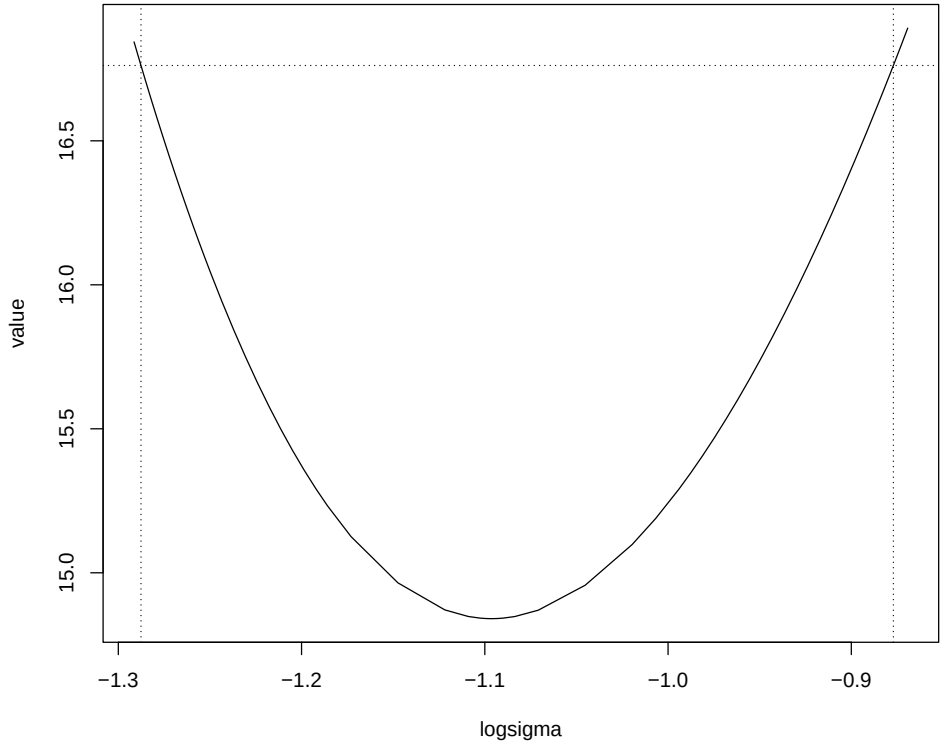
```
ls.pro<-TMB:::tmbprofile(obj, "logsigma")
plot(ls.pro)
```

Profile likelihood

- 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)


#
```

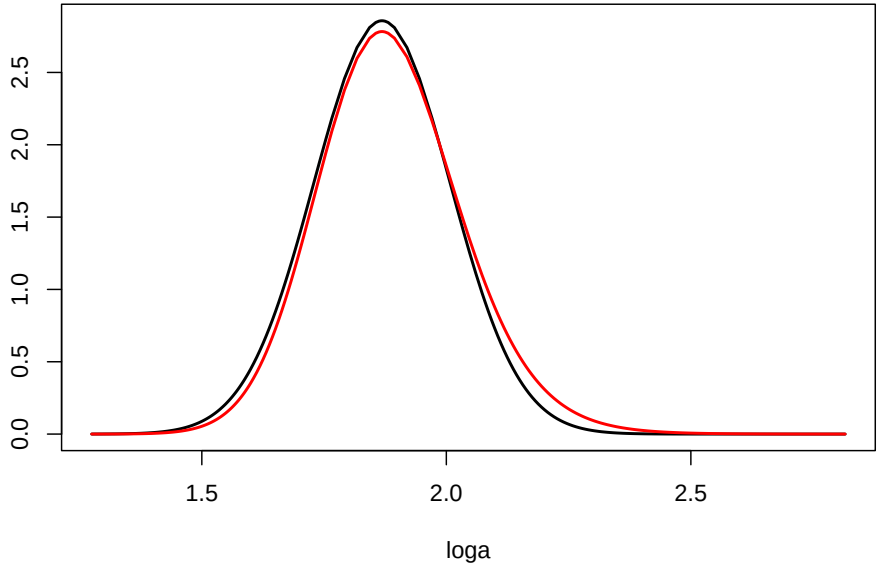
```
               lower      upper
logsigma -1.287628 -0.8770576
```



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:

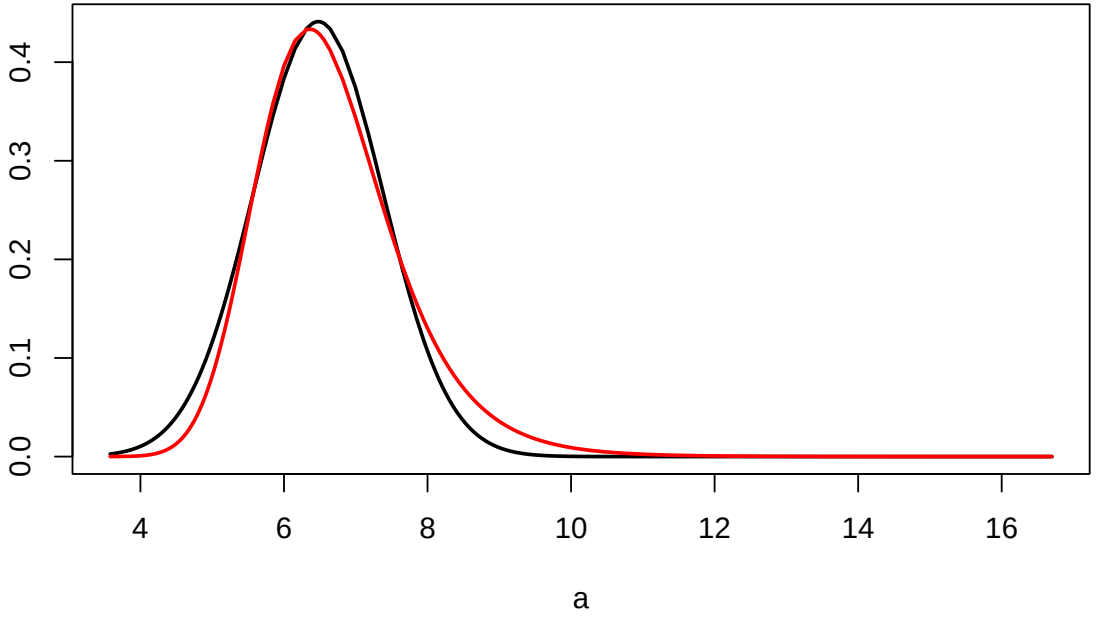
```
density.tmbprofile<-function(x,...){
  xx<-x[,1]
  yy<-exp(-x[,2])
  A<-sum(diff(xx) * (yy[-length(yy)]+yy[-1]))/2
  return(list(x=xx,y=yy/A, data.name=names(x)[1]))
}
pro <- TMB:::tmbprofile(obj,"loga",ytol=10)
sdrep<-summary(sdreport(obj))
plot(pro[,1],dnorm(pro[,1],sdrep["loga",1], sdrep["loga",2]),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



Profile likelihood (cont.)

- How about for a itself?

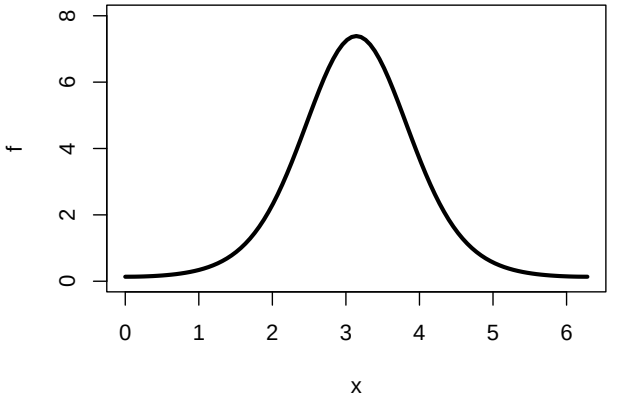
```
pro <- TMB:::tmbprofile(obj,"loga",ytol=10)
d<-density(pro)
pro[,1]<-exp(d$x)
pro[,2]<-d$y/exp(d$x)
plot(pro[,1],dnorm(pro[,1],sdrep["a",1], sdrep["a",2]),
     type="l", lwd=2, xlab="a", ylab="")
lines(pro, lwd=2, col="red")
```



Monte Carlo integration (a toy example).

- Consider the function

$$f(x) = e^{2\cos(x-\pi)}$$



- Calculate the integral:

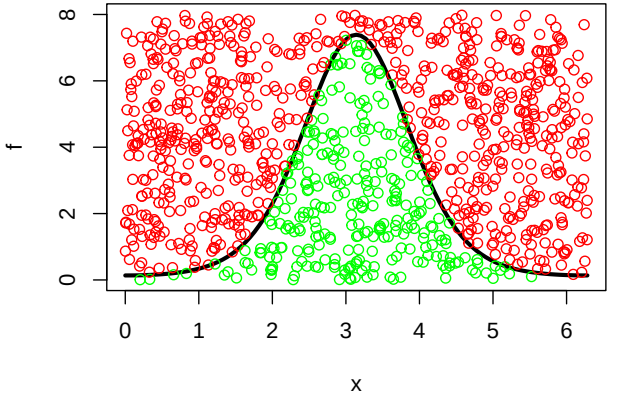
$$\int_0^{2\pi} f(x)dx$$

- A Monte Carlo method is:
  1. Simulate random uniformly distributed numbers $(x_1, y_1), \ldots, (x_N, y_N)$ in the interval $[0, 2\pi] \times [0, 8]$.
  2. Estimate the probability of a point below the curve f by:

$$\widehat{P}_u = \frac{\#\text{below curve}}{N}$$

  3. Area of entire rectangle is 16π, so the estimate of the integral is:

$$\int_0^{2\pi} f(x)dx \approx 16\pi \cdot \widehat{P}_u$$



- Let's try it

Why does it work?

- **Strong law of large numbers (SLLN):** If $X_1, X_2, \ldots$ are independent identically distributed random variables with $E|X| < \infty$, then:

$$\frac{1}{n}\sum_{i=1}^{n} X_i \overset{\text{a.s.}}{\to} E(X), \text{ as } n \to \infty$$

  So it converges to the right thing (subject to very week conditions)

- **Central limit theorem (CLT):** If $X_1, X_2, \ldots$ are independent identically distributed random variables with $EX^2 < \infty$, then:

$$\sqrt{n}\frac{\frac{1}{n}\sum X - E(X)}{\sqrt{\text{var}(X)}} \overset{\mathcal{D}}{\to} N(0,1) \text{ , as } n \to \infty$$

  So a 95% confidence interval is given by:

$$\left|\frac{1}{n}\sum X - E(X)\right| \leq \frac{1}{\sqrt{n}}\text{sd}(X)$$

The convergence rate is proportional to $\frac{1}{\sqrt{n}}$ independent of the dimension.

Other numerical methods have convergence rates proportional to $\frac{1}{\sqrt[d]{n}}$.

Detour: Bayesian inference

- Purely likelihood based inference (a.k.a. Frequentist inference) is based on drawing information from data Y about the model parameters θ via the likelihood function:

$$L(Y|\theta)$$

- In Bayesian inference prior beliefs about the model parameters are expressed as a probability density, so we have:

$$L(Y|\theta) \qquad \text{and} \qquad q(\theta|\psi)$$

- Inference about the model parameters are drawn from the posterior density:

$$p(\theta|Y=y) = \frac{L(Y=y|\theta)q(\theta|\psi)}{\int L(Y=y|\theta)q(\theta|\psi)d\theta}$$

  which is computed via Bayes' rule.

Detour: Bayesian inference

- What is done here is to update the prior beliefs with data
- If the data part is dominating results close to likelihood inference can be expected
- Notice that the prior parameters ψ are not influenced by data. In hierarchical/mixed/random effects models we would estimate those.
- Notice that the prior assumption is entirely subjective and not subject to model validation. In hierarchical/mixed/random effects models we can - to some extend - validate our assumed distribution.

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

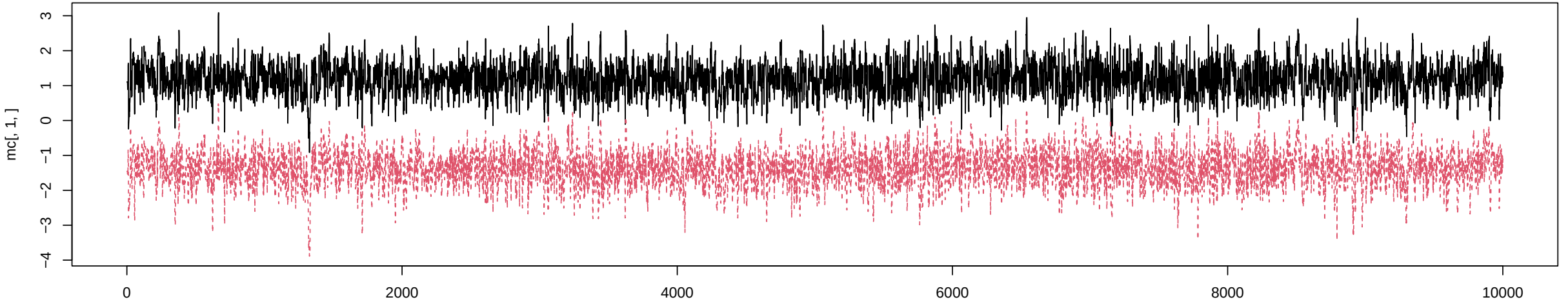
- Assume that these 15 numbers follow a negative binomial distribution:
  13, 5, 28, 28, 15, 4, 13, 4, 10, 17, 11, 13, 12, 17, 3

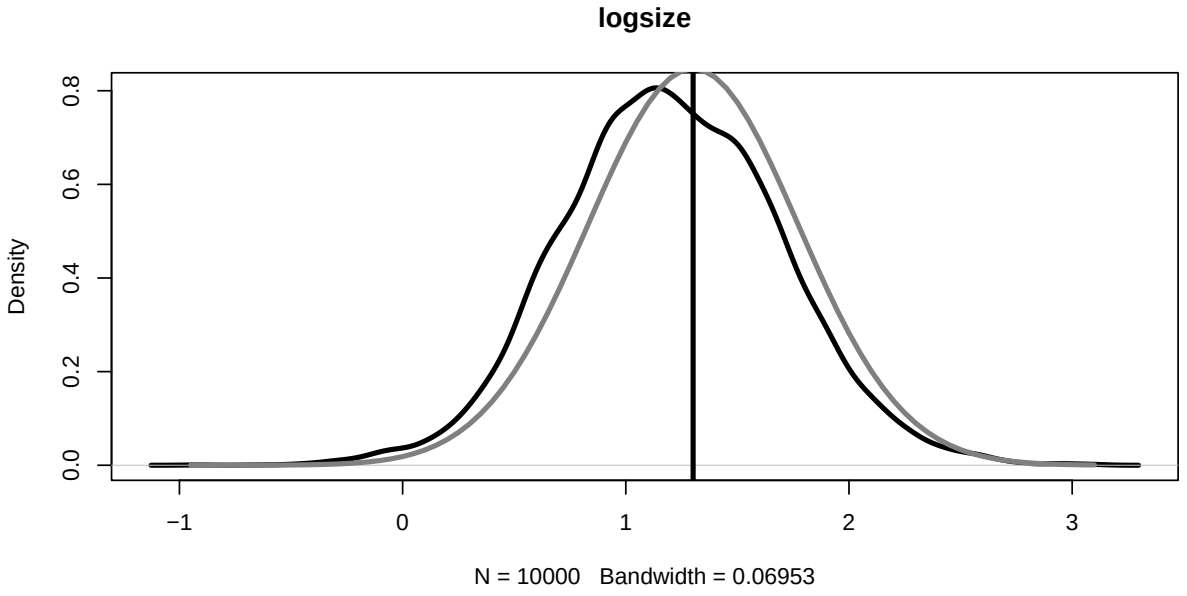
```
library(RTMB)
dat <- list()
dat$Y <- c(13, 5, 28, 28, 15, 4, 13, 4, 10, 17, 11, 13, 12, 17, 3)

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(-5,-5), upper=c(5,5) )
```

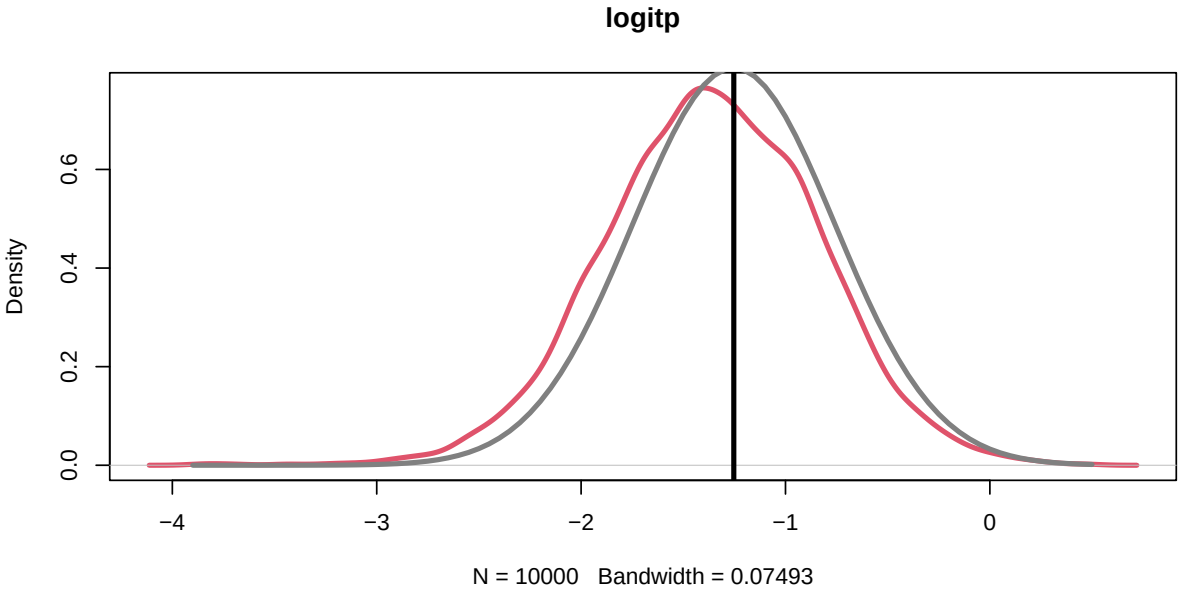
files/mcmc1.R

```
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
−4
0
2000
4000
6000
8000
10000


logsize
Density
0.8
0.6
0.4
0.2
0.0
−1
0
1
2
3
N = 10000 Bandwidth = 0.06953


logitp
Density
0.6
0.4
0.2
0.0
−4
−3
−2
−1
0
N = 10000 Bandwidth = 0.07493

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 θ_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 θ_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

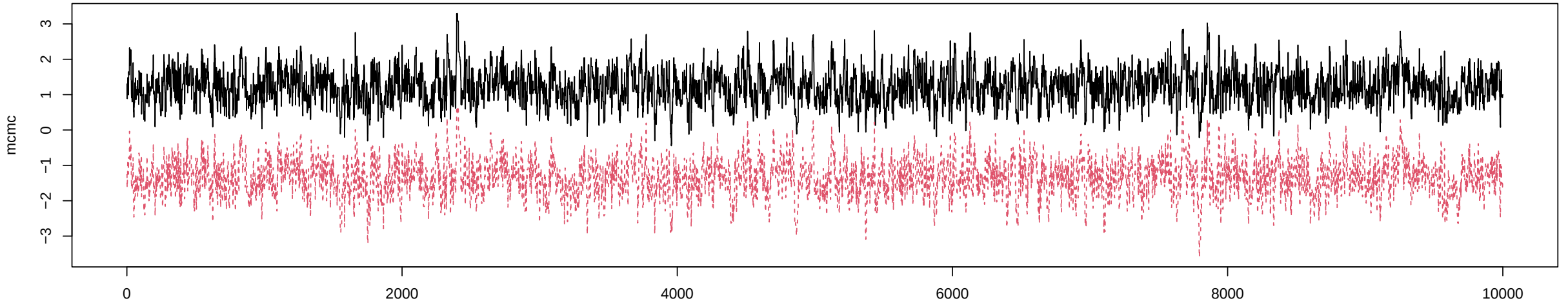
Example: The negative binomial example manually

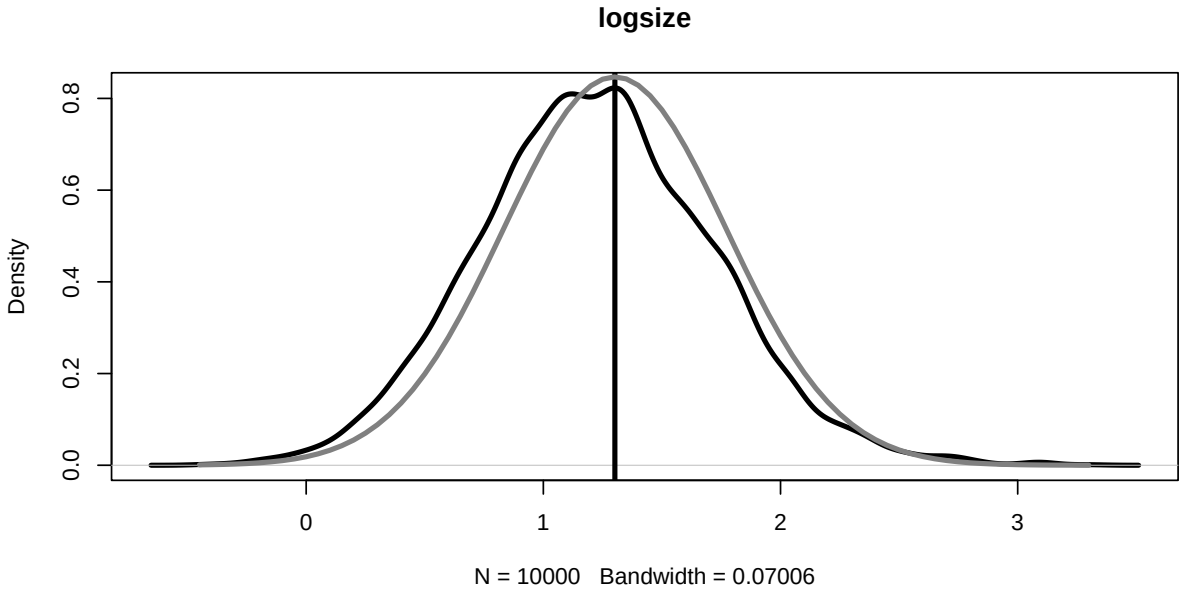
```r
library(RTMB)
dat <- list(Y=c(13,5,28,28,15,4,13,4,10,17,11,13,12,17,3))
par <- list(logsize=0, logitp=0)

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)
```

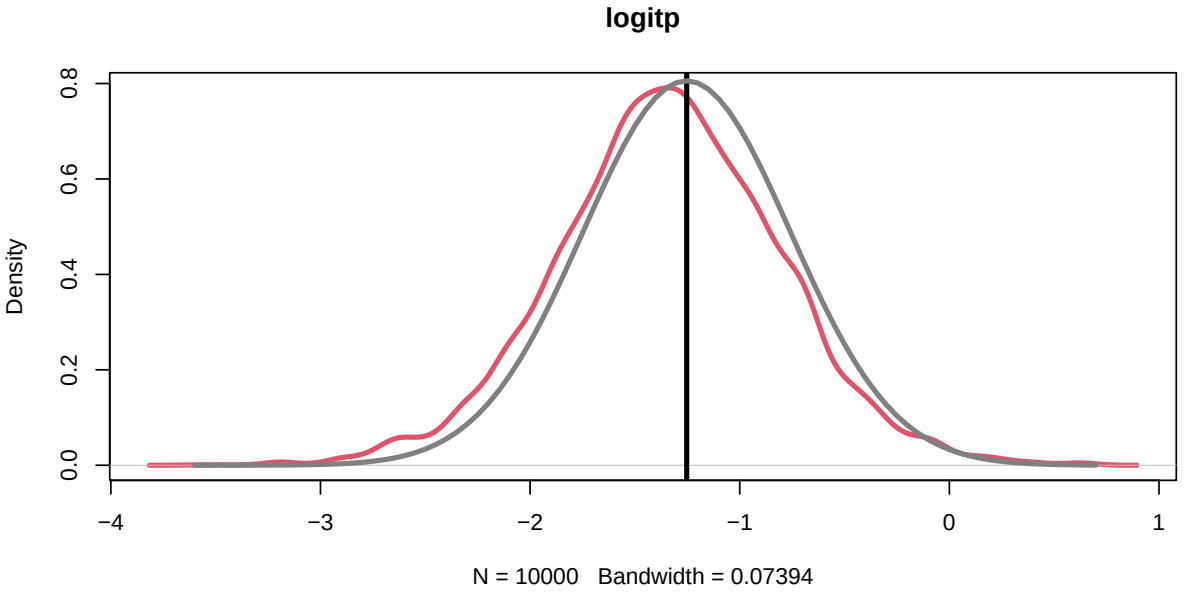
files/mcmc2.R

```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
0.8
0.6
0.4
0.2
0.0
0
1
2
3
N = 10000 Bandwidth = 0.07006


logitp
Density
0.8
0.6
0.4
0.2
0.0
−4
−3
−2
−1
0
1
N = 10000 Bandwidth = 0.07394

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 comparing sequential Bayesian & 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`

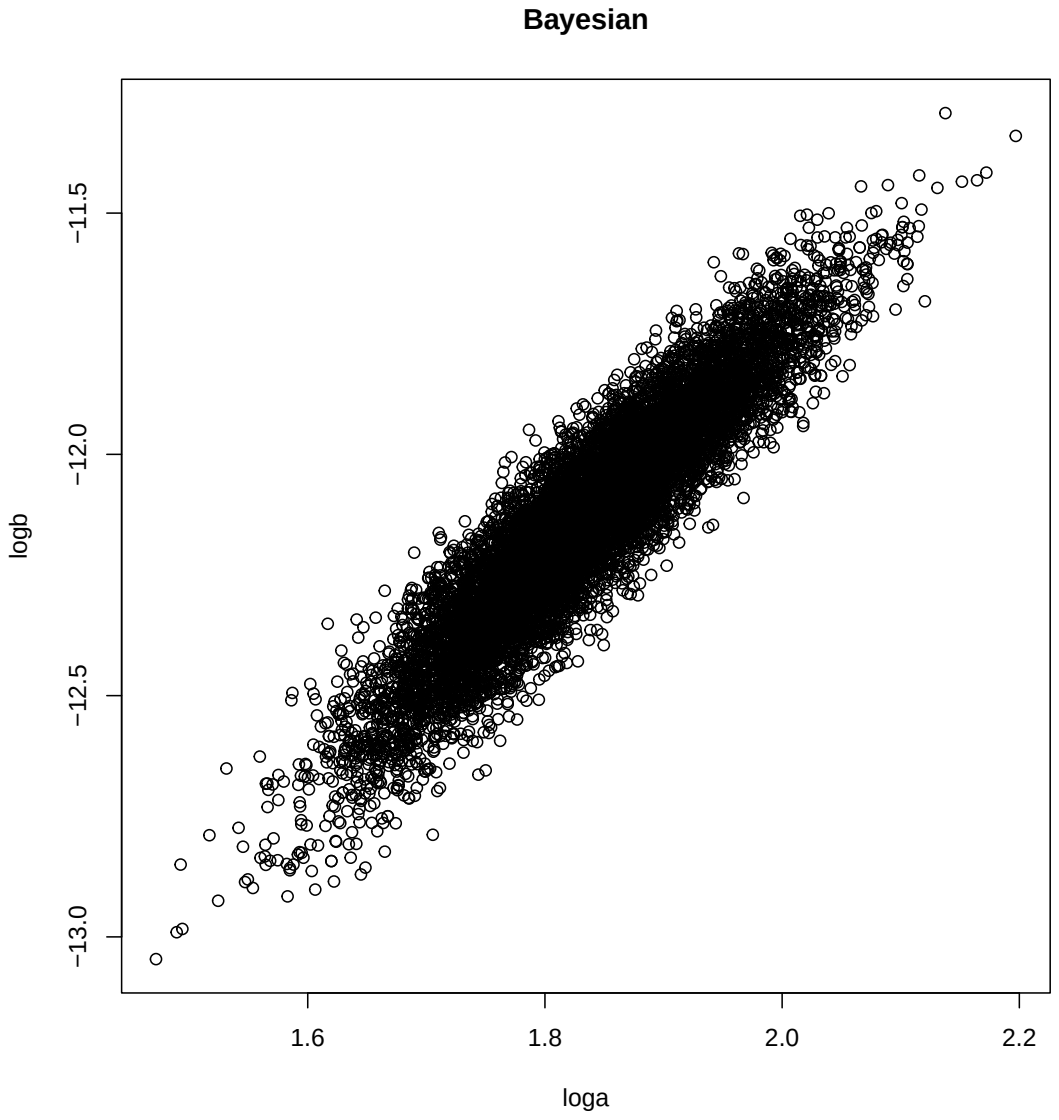
b) Integrated analysis

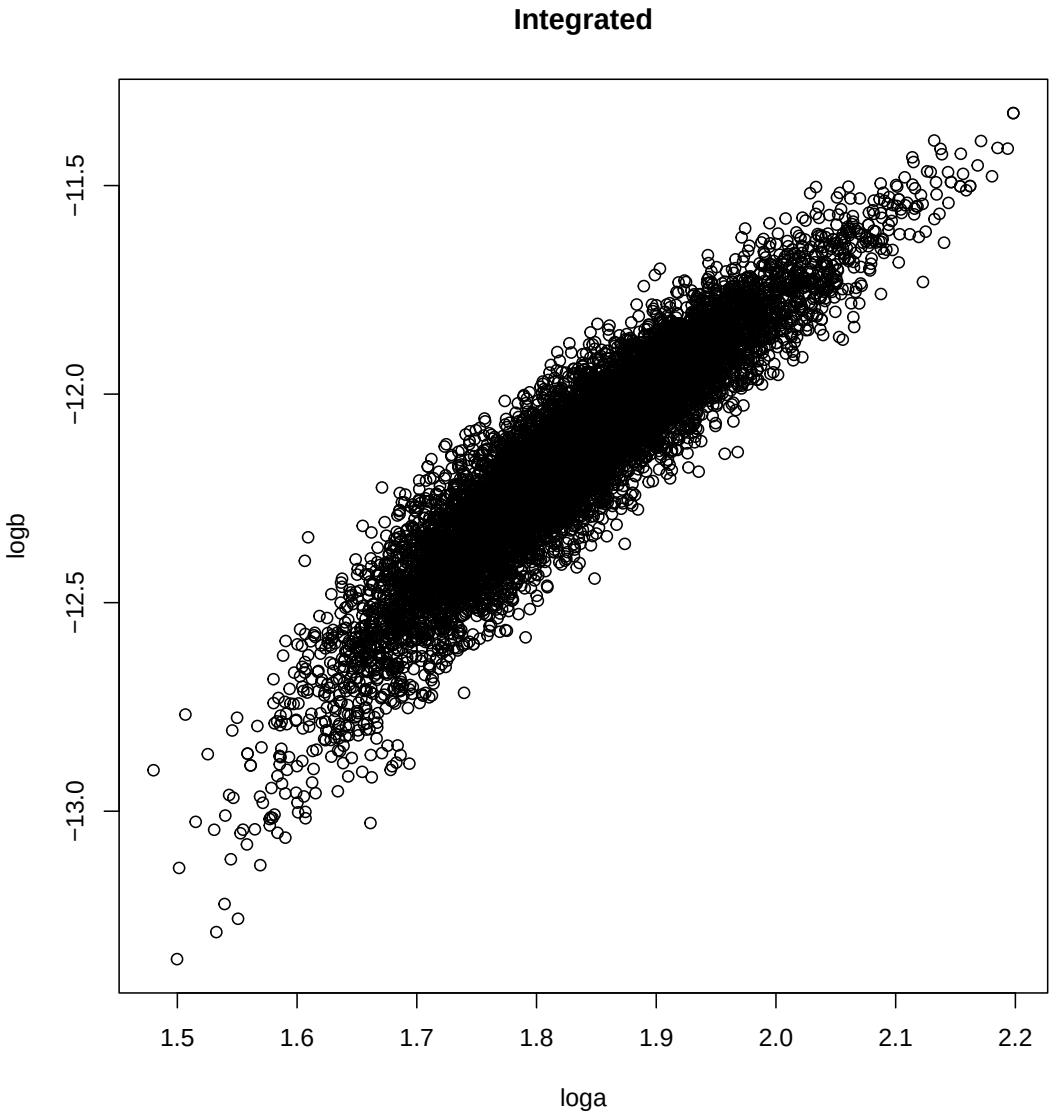
- In the file `bh0.dat` we have the entire data set from the similar stock.
- Modify the program program `bh.R` and to use the both data sets to estimate $\log(a)$ and $\log(b)$.

b) 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





MCMC for the assessment model

- Can we do MCMC for the assessment model?
- Let's try to find the distribution of the last years SSB