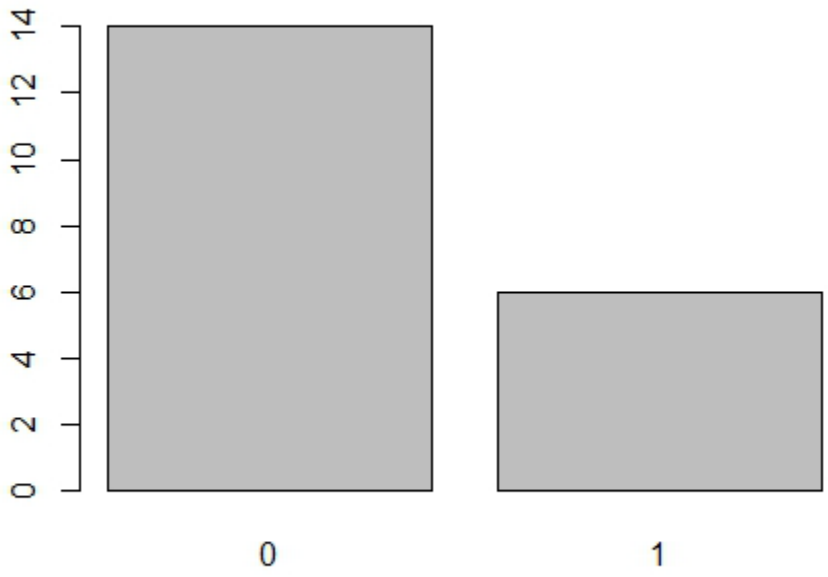
# Exercises

## Chapter 3

1. Use a physical process (i.e., not simulated) to generate twenty Bernoulli observations with probability of success 0.25. Include in your code comments describing how you obtained the data. Produce a summary plot of the total number of 0s and 1s. How does your observed number of successes compare to what you would expect?

```
rm(list=ls()) # Clear Environment

# Bernoulli trials. Draw card with replacement. Heart=1, otherwise 0
Hearts <- c(0,0,0,0,0,0,0,0,0,1,1,0,0,1,0,1,0,0,1,1)
N.obs <- length(Hearts)
Counts <- table(Hearts) # Summary table of 0 vs 1
barplot(Counts)
```

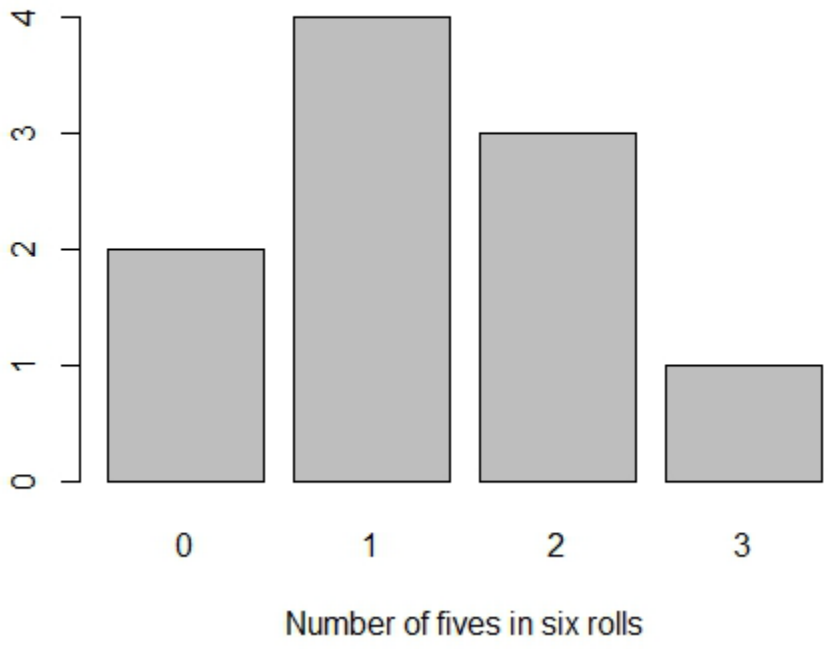
Drew cards with replacement. Hearts were success; otherwise failure. Six of 20 is close to expected number (five).




2. Use a physical process to generate ten binomially distributed observations, with a trial size of at least four. What is the probability of success and the expected number of successes per trial? Produce a histogram showing the frequencies for different numbers of success per trial.

```
rm(list=ls()) # Clear Environment

# Trial is six rolls of a die. How many out of six were fives?
Roll.5 <-c(1,1,0,3,1,2,2,0,2,1)
N.trials <- length(Roll.5) # Function for getting number of trials
Counts <- table(Roll.5)
barplot(Counts)
barplot(Counts, xlab="Number of fives in six rolls")
```

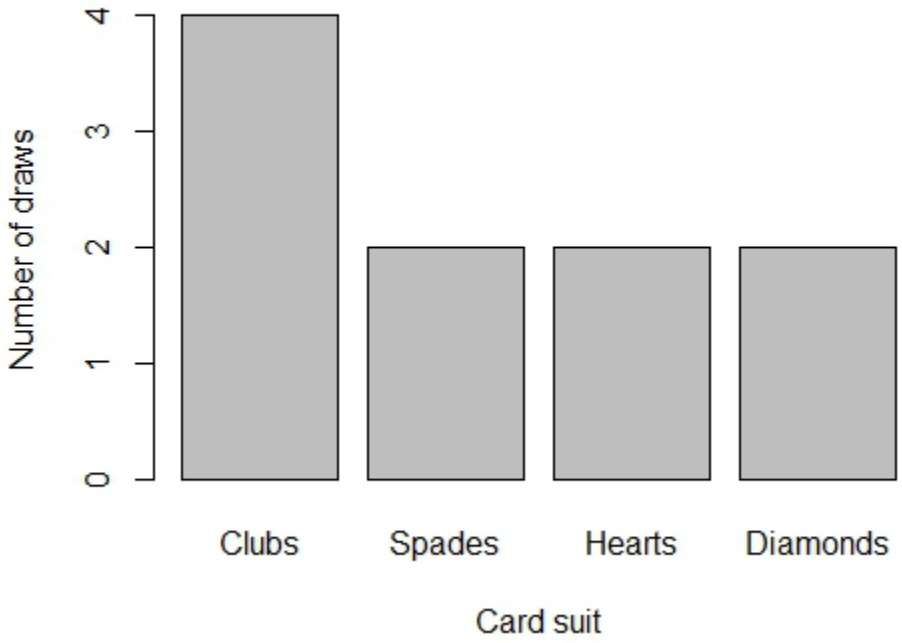
Binomial trial was six rolls of a die. Success was rolling a five. Probability of success was 1/6. Expected number of successes in a trial was 1.




3.Use a physical process to generate 10 outcomes from a multinomial process with at least three possible outcomes. Describe in your code comments the physical process for obtaining the data and include a plot of the frequencies. How do the frequencies compare to what you would expect?

```
rm(list=ls()) # Clear Environment

# Draw ten cards with replacement. Number by suit.
x <-c(4,2,2,2)
Num.Trials <- sum(x)
x.labels <- c("Clubs", "Spades", "Hearts", "Diamonds") # x axis labels
barplot(x, names.arg=x.labels, xlab="Card suit", ylab="Number of
draws")
```

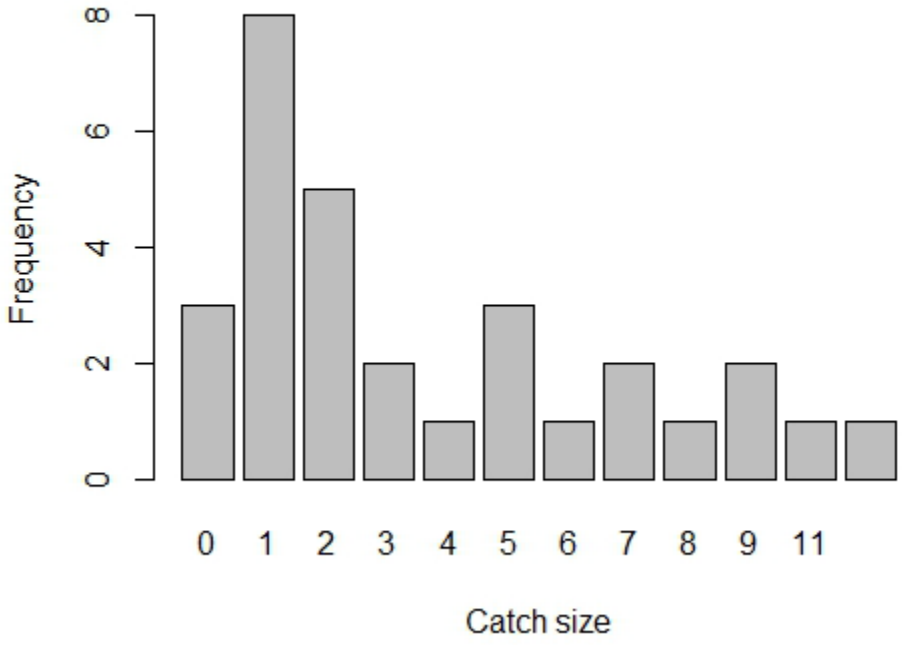
Ten cards drawn with replacement and suit of each card recorded. Expected frequency per suit is 2.5.




4.Assume that the following vector contains trawl catches: y=c(7, 5, 1, 5, 0, 3, 8, 2, 14, 5, 0, 6, 9, 1, 2, 2, 1, 1, 0, 9, 2, 2, 4, 11, 1, 1, 7, 3, 1, 1). Based on your analysis (including plot), were these data likely generated from a Poisson or negative binomial distribution?

```
rm(list=ls()) # Clear Environment

Catch=c(7, 5, 1, 5, 0, 3, 8, 2, 14, 5, 0, 6, 9, 1, 2, 2,
        1, 1, 0, 9, 2, 2, 4, 11, 1, 1, 7, 3, 1, 1)
N <- length(Catch)
Freq <- table(Catch) # Distribution of trawl catches
barplot(Freq, main="", xlab="Catch size", ylab="Frequency")
mean(Catch)
var(Catch)
```

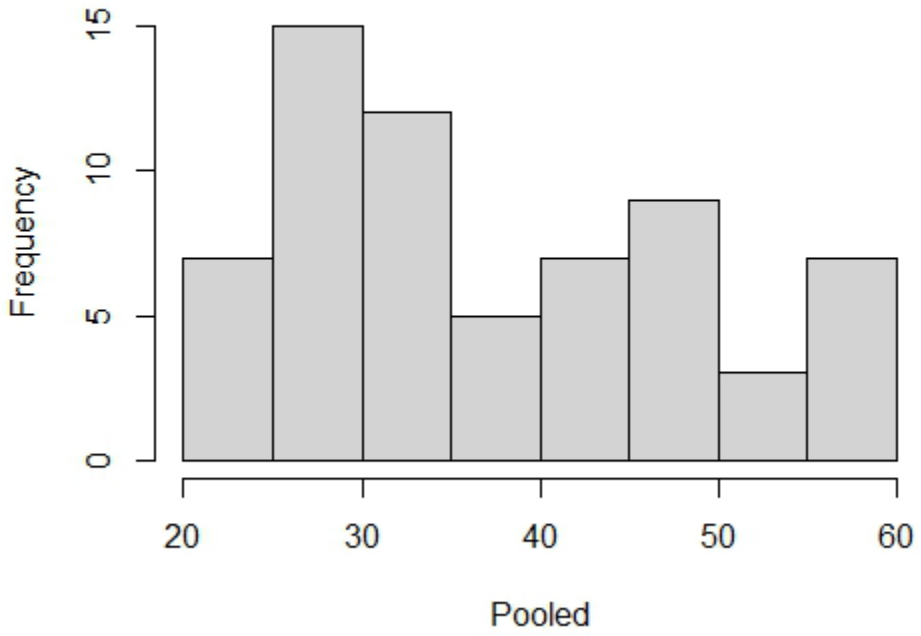
Conclude that negative binomial much more likely. Skewed distribution, variance (13.13)>> mean (3.8).




5. Provide simulation code for generating normally-distributed length data for fish ages 1 (n.1=40, age-1 mean 30, sd 5) and 2 (n.2=25, age-2 mean=50, sd=7). Plot the combined length sample. In multiple simulations using these default values, how frequently is the length distribution visibly bimodal?

```
rm(list=ls()) # Clear Environment

n.1 <- 40 # Sample size for age 1
Len.1 <- rnorm(n=n.1, mean=30, sd=5)
n.2 <- 25 # Sample size for age 2
Len.2 <- rnorm(n=n.2, mean=50, sd=7)
Pooled <- c(Len.1, Len.2)
hist(Pooled, main="")
```

Moderate evidence of bimodality in roughly half of simulations (7 of 12).




## Chapter 4

1. Estimate uncertainty around a two-sample mark-recapture estimate N.hat using the simulation and Bayesian approaches for a capture probability of 0.4. How similar are the results and how do they compare to results using the original capture probability of 0.2?

```
# Simulation
rm(list=ls()) # Clear Environment

# Extended model to include stochasticity
N.true <- 400  # Arbitrary assumed population size
p.true <- 0.4 # Capture probability (fraction of population sampled)
n1 <- N.true * p.true # Caught and marked in first sample
n2 <- N.true * p.true # Caught and examined for marks in second sample

Reps <- 1000
m2.vec <- rbinom(n=Reps, prob=n1/N.true, size=n2)
Nhat.vec <- n1 * n2 / m2.vec

hist(m2.vec, xlab="Number of recaptures", main="")
hist(Nhat.vec, xlab="Population estimate", main="")

mean(Nhat.vec)
quantile(Nhat.vec, probs=c(0.025, 0.5, 0.975), na.rm=TRUE)
# quantile function provides empirical confidence bounds and median

# Bayesian, using lognormal prior for N.hat
```

```
rm(list=ls()) # Clear Environment

# Arbitrary 'observed' values for analysis
N.true <- 400  # Population size
p.true <- 0.4 # Capture probability (fraction of population sampled)
n1 <- N.true * p.true # Caught and marked in first sample
n2 <- N.true * p.true # Caught and examined for marks in second sample
m2 <- n2 * p.true # Marked fish in second sample

# Load necessary library packages
library(rjags)   # Package for fitting JAGS models from within R
library(R2jags)  # Package for fitting JAGS models. Requires rjags

# JAGS code
sink("TwoSampleCR.txt")
cat("
    model {

  # Prior
    N.hat ~ dlnorm(0, 1E-6) # uninformative prior (N.hat>0)
    MarkedFraction <- n1/N.hat

  # Likelihood
    # Binomial distribution for observed recaptures
    m2 ~ dbin(MarkedFraction, n2)
    }

    ",fill = TRUE)
sink()

# Bundle data
jags.data <- list("n1", "n2", "m2")

# Initial values.
jags.inits <- function(){ list(N.hat= rlnorm(n=1, meanlog=10,
sdlog=1))}

model.file <- 'TwoSampleCR.txt'

# Parameters monitored
jags.params <- c("N.hat", "MarkedFraction")

# Call JAGS from R
jagsfit <- jags(data=jags.data, inits=jags.inits, jags.params,
                n.chains = 3, n.thin = 1, n.iter = 2000, n.burnin =
1000,
                model.file)
print(jagsfit, digits=3)
plot(jagsfit)
```

Results

| Simulation | Bayesian |
|---|---|
| 337-483 | 338-499 |
| 337-492 | 338-492 |
| 336-492 | 338-493 |
| 332-492 | 337-496 |
| 341-492 | 340-494 |

Percentiles similar for the two approaches. Considerably narrower intervals than for capture probability of 0.2 (286-43139, 275-699, 269-652).

2. @robson.regier1964 provided practical advice on acceptable levels of accuracy and precision. They suggested that estimates within 10% of the true value be acceptable for research studies, versus 25% for management and 50% for preliminary studies. Bayesian credible intervals could be used as a proxy for their accuracy and precision targets. What capture probabilities (approximate) would correspond to those targets for the mark-recapture example?

For population size of 400, 10% interval would be 360-440; 25% would be 300-500, and 50% would be 200-600. Use same code as Bayesian case in exercise #1.

| Capture prob. | Lower | Upper | Width |
|---|---|---|---|
| 0.1 | 194 | 1719 | 1525 |
| 0.15 | 241 | 888 | 647 |
| 0.2 | 279 | 673 | 394 |
| 0.3 | 314 | 555 | 241 |
| 0.4 | 340 | 494 | 154 |
| 0.5 | 352 | 465 | 113 |
| 0.6 | 365 | 449 | 84 |

Comparing the width of the interval, 10% (width 80) is closest to a 0.6 capture probability, 25% (width 200) closest to 0.3, and 50% (width 400) closest to 0.2.

3. How many errors can you locate before running the following code? Fix all errors that you spotted then run the code to fix any remaining errors.

I count six errors (red text color below). Note that MaxCount could have been passed in the jags.data list, but here I just replaced it with the fixed value in the JAGS code.

```
rm(list=ls()) # Clear Environment

# Load necessary library packages
library(rjags)   # Package for fitting JAGS models from within R
library(R2jags)  # Package for fitting JAGS models. Requires rjags

# Arbitrary 'observed' values for analysis
Reps <- 20
AveC <- 3
Count <- rpois(n=Reps, lambda=AveC)
```

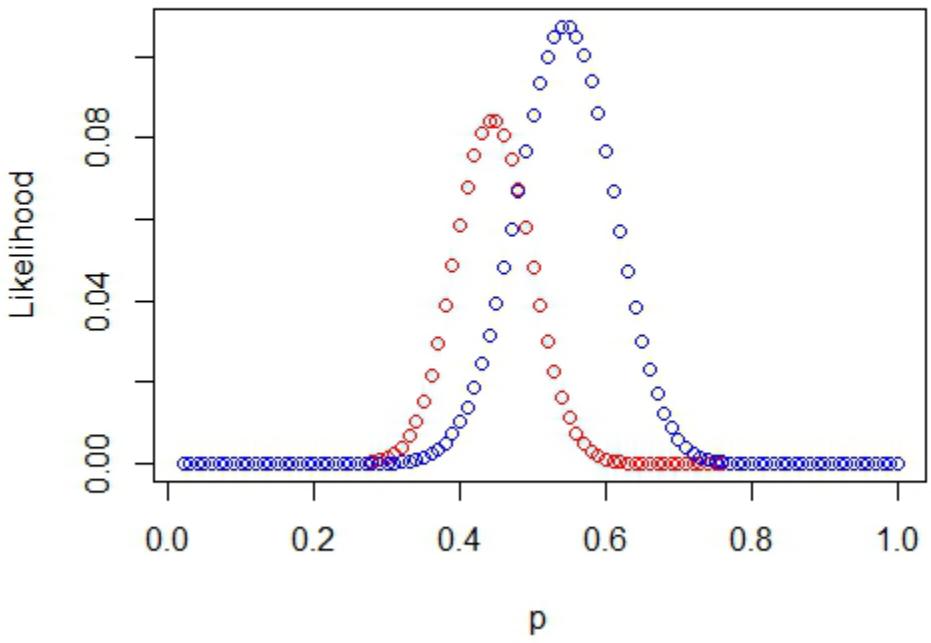
```
# JAGS code
sink("PoissonSim.txt")
cat("
    model {

    # Priors
    AveC.est ~ dunif(0, 100)

    # Likelihood
    for (i in 1:Reps){
    Count[i] ~ dpois(AveC.est)
    } #i
}
    ",fill = TRUE)
sink()

# Bundle data
jags.data <- list("Count", "Reps")

# Initial values.
jags.inits <- function(){ list(AveC.est=runif(n=1, min=0, max=100))}

model.file <- 'PoissonSim.txt'

# Parameters monitored
jags.params <- c("AveC.est")

# Call JAGS from R
jagsfit <- jags(data=jags.data, inits=jags.inits, jags.params,
                n.chains = 3, n.thin = 1, n.iter = 2000, n.burnin =
1000,
                model.file)
print(jagsfit, digits=3)
plot(jagsfit)
```

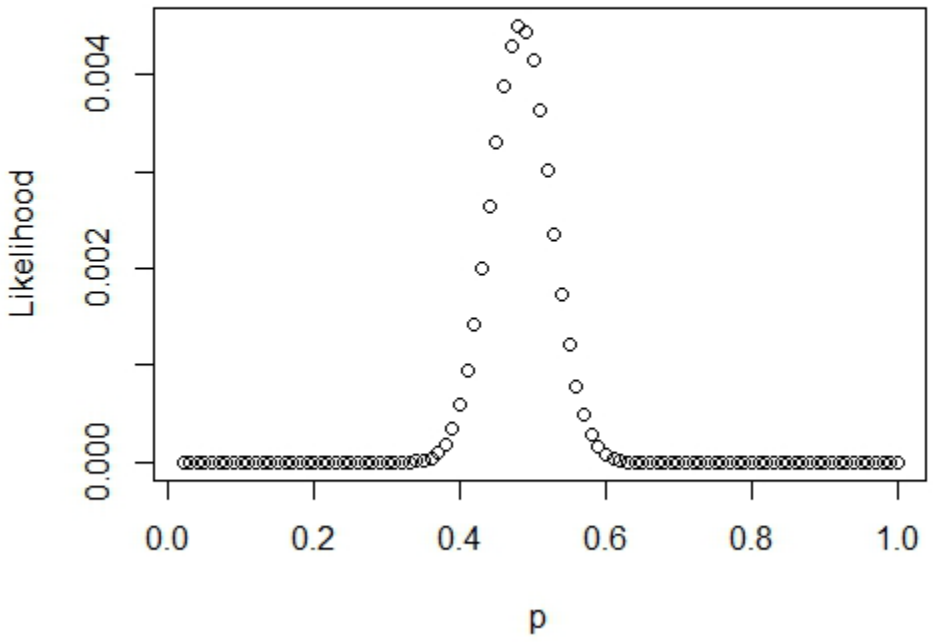
4. Consider a mark-recapture study with n1 = 60 and two independent samples for recaptures (n2=90, m2=40; n3=55, m3=30). Use a vector for p at 0.01 intervals and produce a vector and plot of joint likelihoods (vector given the name "both"). Assume that only the initial sample (n1) is marked; samples n2 and n3 provide two independent snapshots of the marked fraction. Explain in detail how p[which.max(both)] works and what it produces.

```
rm(list=ls()) # Clear Environment

# Mark-recapture example

n1 <- 60 # Caught and marked in first sample
n2 <- 90 # Caught and examined for marks in second sample
n3 <- 55 # Caught and examined for marks in third sample
m2 <- 40 # Marked fish in second sample
m3 <- 30 # Marked fish in second sample
```

```
# Create vector of possible p values, determine likelihood for each
p <- seq(from=0.02, to=1, by=0.01)
l2.vec <- choose(n2,m2) * p^m2 * (1-p)^(n2-m2)
l3.vec <- choose(n3,m3) * p^m3 * (1-p)^(n3-m3)
par(mfrow=c(1,1)) # Reset plot frame
yrange <- c(0, max(l2.vec, l3.vec))
plot(p, l2.vec, ylab="Likelihood", col="red", ylim=yrange)
points(p, l3.vec, col="blue")
both <- l2.vec*l3.vec
plot(p, both, col="black", ylab="Likelihood")
p[which.max(both)]
```

The y-axis range is greater for l3.vec than l2.vec so a common range was determined before creating the first plot. The which.max() function determines the position within the both array at which the maximum occurs, and returns the value for p at that array location (p=0.48). This value is intermediate between the point estimates from sample 2 (0.44) and 3 (0.545), but closer to the sample 2 point estimate because that sample size is larger.






5. Use the approach from Section 4.3.7 to compare model fits for a more "Poisson-like" negative binomial distribution (e.g., k=10).

```
# DIC comparison for model selection, using "observed" data
# from a negative binomial distribution

rm(list=ls()) # Clear Environment

N <- 30
mu <- 7
k=50
variance <- mu+(mu^2)/k
Count <- rnbinom(n=N, mu=mu, size=k) # Drawn from negative binomial
Freq <- table(Count)  # Distribution of simulated counts
barplot(Freq, main="", xlab="Count", ylab="Frequency")

# Load necessary library
library(rjags)
library(R2jags)

sink("PoissonFit.txt")
cat("
model {

# Prior
```

```
 lambda.est ~ dunif(0, 100)

# Likelihood
    for(i in 1:N) {
      Count[i] ~ dpois(lambda.est)
      } #y
}
    ",fill=TRUE)
sink()

# Bundle data
jags.data <- list("N", "Count")

# Initial values
jags.inits <- function(){ list(lambda.est=runif(n=1, min=0, max=100))}

model.file <- 'PoissonFit.txt'

# Parameters monitored
jags.params <- c("lambda.est")

# Call JAGS from R
jagsfit <- jags(data=jags.data, jags.params, inits=jags.inits,
                n.chains = 3, n.thin=1, n.iter = 10000,
                model.file)
print(jagsfit)
#plot(jagsfit)

Poisson.DIC <- jagsfit$BUGSoutput$DIC

sink("NBFit.txt")
cat("
model {

# Prior
 p.est ~ dunif(0, 1)
   # JAGS uses p (probability of success) for negative binomial
 k.est ~ dunif(0, 1000)
 mu.est <- k.est*(1-p.est)/p.est

# Likelihood
    for(i in 1:N) {
      Count[i] ~ dnbinom(p.est, k.est)
      } #y
}
    ",fill=TRUE)
sink()

# Bundle data
jags.data <- list("N", "Count")

# Initial values
```

```
jags.inits <- function(){ list(p.est=runif(n=1, min=1E-6, max=1),
                               k.est=runif(n=1, min=0, max=1000))}

model.file <- 'NBFit.txt'

# Parameters monitored
jags.params <- c("mu.est", "k.est")

# Call JAGS from R
jagsfit <- jags(data=jags.data, jags.params, inits=jags.inits,
                n.chains = 3, n.thin=1, n.iter = 10000,
                model.file)
print(jagsfit)
#plot(jagsfit)
NB.DIC <- jagsfit$BUGSoutput$DIC
Poisson.DIC - NB.DIC
```

Difference in DIC (Poisson-NB) for k=50:
0.27, 0.36, -0.88, 0.93, -0.31

Difference in DIC (Poisson-NB) for k=10:
1.59, 5.34, 1.67,1.15, 1.10

## Chapter 5

1. 1. Create a table of estimated credible intervals for population size, using several replicate simulations and a range of values for capture probability. What are the lower limits below which estimates are judged not to be reliable?

Code is unmodified from book, except for using different values for capture probability.

```
# Load necessary library
library(rjags)   # Package for fitting JAGS models from within R
library(R2jags)  # Package for fitting JAGS models. Requires rjags

# JAGS code for fitting model
sink("RemovalModel.txt")
  cat("
model{

# Priors
 CapProb.est ~ dunif(0,1)
 N.est ~ dunif(TotalCatch, 2000)

 N.remaining[1] <- trunc(N.est)
 for(j in 1:N.removals){
      Catch[j]~dbin (CapProb.est, N.remaining[j]) # jth removal
      N.remaining[j+1] <- N.remaining[j]-Catch[j] # Remaining
population after removal
  } #j
```

```
}
      ",fill=TRUE)
  sink()

     # Bundle data
  jags.data <- list("Catch", "N.removals", "TotalCatch")

  TotalCatch <- sum(Catch[])

  # Initial values
  jags.inits <- function(){ list(CapProb.est=runif(1, min=0, max=1),
                                 N.est=runif(n=1, min=TotalCatch,
max=2000))}

  model.file <- 'RemovalModel.txt'

  # Parameters monitored
  jags.params <- c("N.est", "CapProb.est")

   # Call JAGS from R
  jagsfit <- jags(data=jags.data, jags.params, inits=jags.inits,
                  n.chains = 3, n.thin = 1, n.iter = 40000,
                  model.file)
  print(jagsfit)
  plot(jagsfit)
```

| Capt. prob. | Replicates | | | | |
|---|---|---|---|---|---|
| 0.1 | 53.8-1766.0 | 41.7-1644.8 | 44.0-1460.6 | 48.3-707.6 | 72.4-1847.0 |
| 0.2 | 83.2-188.4 | 75.0-188.0 | 77.7-592.7 | 79.1-210.7 | 77.9-1525.4 |
| 0.3 | 86.4-156.6 | 89.6-215.4 | 92.6-148.5 | 91.6-180.2 | 89.6-135.3 |
| 0.4 | 87.8-102.1 | 90.3-107.7 | 95.4-127.3 | 91.4-109.3 | 94.9-108.6 |

Results for 0.1 would not be useful. Results for 0.2 typically would be, and there is further improvement at 0.3. Estimates at 0.4 are quite precise, if a capture probability that high could be achieved.

2. For a two-sample mark-recapture experiment with a true population size of 1000, produce a table of credible intervals for N.hat for different assumed capture probabilities. What sampling intensity would produce an estimate with uncertainty close to the Robson and Regier target for management (+/- 25%)?

| Capt. prob. | Replicates | | | | |
|---|---|---|---|---|---|
| 0.1 | 619.0-2032.5 | 506.9-1405.8 | 532.9-1564.9 | 572.2-1733.6 | 717.1-2805.0 |
| 0.2 | 894-4-1650.2 | 799.2-1396.9 | 668.4-1098.4 | 661.1-1077.6 | 628.3-1000.2 |
| 0.3 | 887.4-1272.0 | 854.4-1214.0 | 1034.8-1577.2 | 830.4-1157.0 | 998.6-1482.7 |
| 0.4 | 929.7-1184.2 | 899.9-1142.2 | 885.4-1118.3 | 900.6-1144.2 | 860.7-1089.1 |

+/-25% is a width of 500 for a population size of 1000. Capture probability of 0.2 roughly comparable width (target limits 750 and 1250; actual limits vary widely).

3. Modify the binomial-mixture example to compare credible intervals for estimated mean abundance (lam.est) using fewer sites but more replicates, preserving the same total number of observations (e.g., 10 sites and 9 replicate survey visits). Can you suggest potential advantages and disadvantages of using fewer sites?

Code same as in book except reduced number of iterations (check Rhat!):

```
rm(list=ls()) # Clear Environment

Sites <- 30
Replicates <- 3
lam.true <- 20 # Mean number per site
p.true <- 0.5 # Detection probability

Counts <- matrix(NA, nrow=Sites, ncol=Replicates)
Site.N <- rpois(n = Sites, lambda = lam.true) # True abundance at each
site
for (i in 1:Sites){
  Counts[i,] <- rbinom(n = Replicates, size = Site.N[i], prob =
p.true)
} #i
head(Site.N)
head(Counts)

# Load necessary library
library(rjags)
library(R2jags)

# Specify model in BUGS language
sink("N-mix.txt")
cat("
model {

# Priors
  lam.est ~ dunif(0, 100)  # uninformative prior for mean abundance at
each site
  p.est ~ dunif(0, 1)

# Likelihood
# Biological model for true abundance
for (i in 1:Sites) {
   N.est[i] ~ dpois(lam.est)
   # Observation model for replicated counts
   for (j in 1:Replicates) {
      Counts[i,j] ~ dbin(p.est, N.est[i])
      } # j
   } # i
totalN <- sum(N.est[])     # Calculated variable: total pop. size
across all sites
}
",fill = TRUE)
```

```
sink()

# Bundle data
jags.data <- list("Counts", "Sites", "Replicates")

# Initial values
Nst <- apply(Counts, MARGIN=1, max) + 1     # Kery and Schaub
jags.inits <- function(){ list(lam.est=runif(n=1, min=0, max=100),
                               N.est=Nst, p.est=runif(1, min=0,
max=1))}

model.file <- 'N-mix.txt'

# Parameters monitored
jags.params <- c("lam.est", "p.est", "totalN")

# Call JAGS from R
jagsfit <- jags(data=jags.data, jags.params, inits=jags.inits,
                n.chains = 3, n.thin=1, n.iter = 50000, # Reduced
iterations from book. Check Rhat!!
                model.file)
print(jagsfit)
par(mfrow=c(1,1)) # Reset plot panel
plot(jagsfit)

# Estimated posterior distribution for total population size
hist(jagsfit$BUGSoutput$sims.list$totalN, xlab="Estimated total pop
size", main="")
abline(v=sum(Site.N), col="red")
```

| Default (30 sites, 3 reps) | 10 sites, 9 reps |
|---|---|
| 19.0-72.7 | 15.0-34.0 |
| 15.9-33.9 | 14.5-28.9 |
| 13.1-20.3 | 16.0-30.4 |
| 14.9-48.5 | 13.8-47.5 |
| 13.1-20.4 | 15.2-35.9 |

Five is a limited number of replicate comparisons but results appear to be similar. Using fewer sites simplifies the logistics, but runs a risk if variability among sites is non-random (e.g., due to environmental variables). Ninety samples from 30 versus 10 sites depends on the sites being true spatial replicates.

## Chapter 6

1. For the default age-composition analysis without gear selectivity (Section 6.1), compare credible intervals for survival rate at sample sizes of 30 versus 200 fish.

| Sample size 30 | Sample size 200 |
|---|---|
| .67-.90 | .66-.75 |
| .57-.80 | .61-.70 |

| | |
|---|---|
| .57-.80 | .69-.78 |
| .59-.81 | .61-.70 |
| .69-.92 | .64-.73 |

Results from the higher sample size are quite precise and should be very useful for management or research.

2. For the telemetry method of estimating survival (Section 6.2), approximately what sample size would produce an estimate with the level of precision recommended by Robson and Regier (1964) for research (10%) studies (50%)?

Code is the same as in the book except for the changed sample size.

| Sample size | Credible interval | Width | Credible interval | Width |
|---|---|---|---|---|
| 20 | .29-.66 | .37 | .49-.80 | .31 |
| 50 | .36-.60 | .24 | .38-.62 | .24 |
| 100 | .45-.60 | .15 | .43-.59 | .16 |
| 200 | .46-.57 | .11 | .49-.59 | .10 |

For a survival rate of 0.5, +/- 10% is a width of 0.1, which is approximately the width for a sample size of 200.

3. For the tag-return method (Section 6.3), compare credible intervals for survival rate at releases of 50, 100, and 200 fish. How do the results compare to the Robson and Regier (1964) recommendations? In what ways is this simulated field study optimistic about the reliability of results?

Code same as book except changed NTags.

| NTags=50 | NTags=100 | NTags=200 |
|---|---|---|
| .47-.98 | .56-.98 | .48-.82 |
| .62-1.00 | .50-.98 | .48-.82 |
| .26-.95 | .62-.96 | .63-.98 |
| .52-.99 | .48-.96 | .63-.98 |
| .52-.99 | .62-.99 | .56-.97 |

+/- 10% is 0.72-0.88, which would require a much larger sample size. 25% is 0.6-1.0, which is similar to the results for 200 tags. The lowest precision case (50%) has a lower bound of 0.4; the upper bound would not make sense for a probability. 0.4 to 1 is roughly similar to the 50 or 100 tag cases.

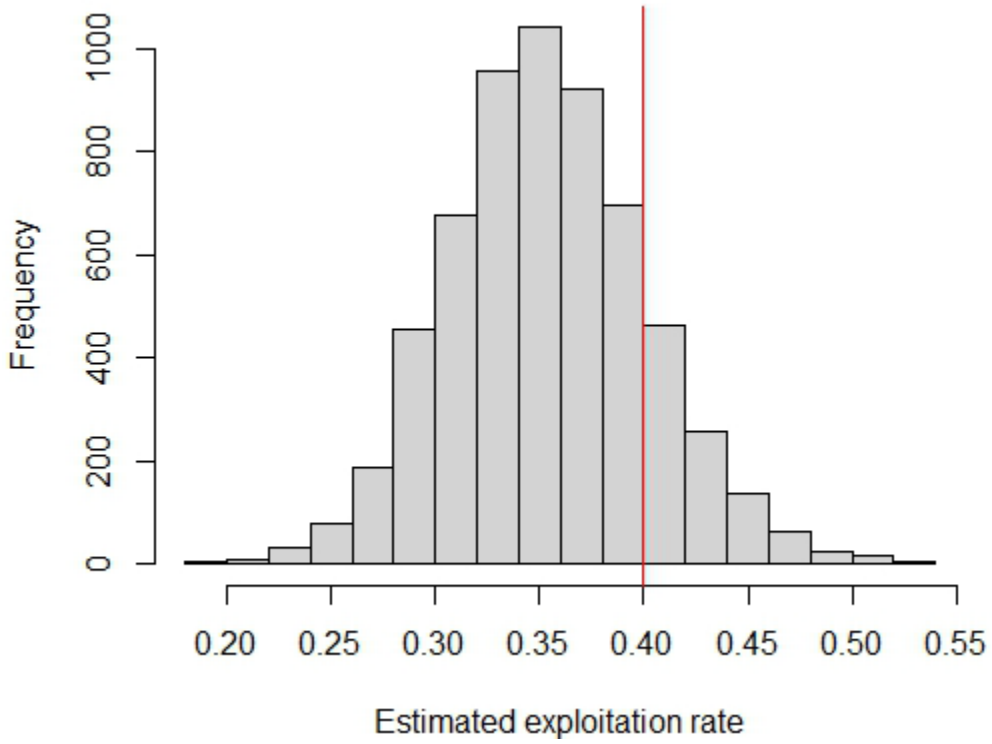
A variety of practical issues would make these results optimistic (tag loss, tagging mortality, sampling that doesn't perfectly conform to the assumed binomial process). Yet even the highest sample size case provides little information about survival rate (e.g., 0.6-1 is not very informative). I would not recommend carrying out a field study using any of these sample sizes.

## Chapter 7

1. For the exploitation rate study design, modify the JAGS code to plot the posterior distribution for the exploitation rate. Include a vertical line showing the underlying true exploitation rate.

```
rm(list=ls()) # Clear Environment

# Tagging study to estimate exploitation rate
```

```
u.true <- 0.4
n.tagged <- 100
n.returned <- rbinom(n=1, prob=u.true, size=n.tagged)

# Load necessary library
library(rjags)
library(R2jags)

# JAGS code
sink("ExpRate.txt")
cat("
model{
    # Priors
    u.est ~ dunif(0,1)  # Exploitation rate

    # Likelihood
    n.returned ~ dbin(u.est, n.tagged)
}
    ",fill=TRUE)
sink()

# Bundle data
jags.data <- list("n.returned", "n.tagged")

# Initial values
jags.inits <- function(){ list(u.est=runif(n=1, min=0, max=1))}

model.file <- 'ExpRate.txt'

# Parameters monitored
jags.params <- c("u.est")

# Call JAGS from R
jagsfit <- jags(data=jags.data, jags.params, inits=jags.inits,
                n.chains = 3, n.thin=1, n.iter = 4000, n.burnin=2000,
                model.file)
print(jagsfit)
plot(jagsfit)

# Look at posterior distribution versus true value
hist(jagsfit$BUGSoutput$sims.list$u.est,
     xlab="Estimated exploitation rate", main="")
abline(v=u.true, col="red")
```




2. For the completed tagging study design, modify the code to use a single F parameter (representing an average over periods).

Simulation code unchanged. Using a single F means that none of the JAGS parameters are time-dependent. Cell probabilities are time-dependent.

```
rm(list=ls()) # Clear Environment

n.tagged <- 1000
Periods <- 40
# Set to high value to ensure that end of study is observed
F.true <- runif(n=(Periods-1), min=0.3, max=0.5)
M.true <- 0.2
S.true <- exp(-F.true - M.true)
u.true <- F.true * (1-S.true) / (F.true + M.true)

TagsAtRisk <- array(data=NA, dim=Periods)
TagsAtRisk[1] <- n.tagged
TagFates <- array(data=NA, dim=(Periods-1)) # Returns by period

for (j in 2:Periods){
  TagsAtRisk[j] <- rbinom(n=1, size=TagsAtRisk[j-1], prob=S.true[j-1])
  TagFates[j-1] <- rbinom(n=1, size=(TagsAtRisk[j-1]
                                     -TagsAtRisk[j]),prob=(F.true[j-1]
                                                           /(F.true[j-
1]+M.true)))
  # Random realization: fishing deaths
```

```
} #j

t.Completed <- max(which(TagFates[]>0))
# Locate period when study completed (last return)
NatDeaths <- n.tagged - sum(TagFates)
TagFates <- c(TagFates[1:t.Completed],NatDeaths)
# Returns + all natural deaths

# Load necessary library packages
library(rjags)
library(R2jags)

# JAGS code
sink("Hearn.txt")
cat("
  model {
  # Priors
  M.est ~ dunif(0, 2)  # Instantaneous natural mortality rate
  F.est ~ dunif(0,2) # Instantaneous fishing mortality rate
     S.est <- exp(-M.est - F.est)
     u.est <- F.est * (1-S.est)/(F.est+M.est) # FA/Z

# Cell probabilities
  p.est[1] <- u.est
  for (i in 2:t.Completed) {
    p.est[i] <- S.est^(i-1)*u.est
      } #i
    p.est[t.Completed+1] <- 1 - sum(p.est[1:t.Completed])
      # Prob of not being seen again
  TagFates[1:(t.Completed+1)] ~ dmulti(p.est[1:(t.Completed+1)],
                                n.tagged)
 }
  ",fill=TRUE)
sink()

# Bundle data
jags.data <- list("TagFates", "t.Completed", "n.tagged")

# Initial values
jags.inits <- function(){list(M.est=runif(1, 0, 2),
                              F.est=runif(1, 0, 2))}

model.file <- 'Hearn.txt'

# Parameters monitored
jags.params <- c("M.est", "F.est")

# Call JAGS from R
jagsfit <- jags(data=jags.data, jags.params, inits=jags.inits,
                n.chains = 3, n.iter = 4000, n.burnin=2000,
                model.file)
print(jagsfit)
```

```
plot(jagsfit)

par(mfrow=c(1,1)) # Default plot settings
mar = c(5.1, 4.1, 4.1, 2.1)
# Look at posterior distribution versus true value
hist(jagsfit$BUGSoutput$sims.list$M.est,
     xlab="Estimated natural mortality rate", main="")
abline(v=M.true, col="red")

plot(1:t.Completed, F.true[1:t.Completed], type="b")
abline(h=jagsfit$BUGSoutput$median$F.est, col="green")
```

Estimates of F are much improved for the reduced model.

3. For the multi-year tagging study, run three replicate simulations of the version using planted tags, with tag releases of 1000 (current code), 100, and 30. How does uncertainty (e.g. credible interval width) vary? What sample size would you recommend if planning this study? Also, save credible intervals for v3 for the 1000 fish releases (for exercise 4 below).

Code unmodifed except for the sample size.

| 1000 u3 | v3 | 100 u3 | v3 | 30 u3 | v3 |
|---|---|---|---|---|---|
| .26-.36 | .02-.69 | .24-.43 | .02-.67 | .17-.52 | .02-.70 |
| .26-.34 | .02-.69 | .24-.43 | .02-.67 | .34-.69 | .01-.55 |
| .25-.35 | .02-.69 | .23-.45 | .02-.68 | .18-.49 | .02-.70 |

More of a change in u3 than v3. Consistent increase in credible intervals as sample size decreases. Sample size of 100 should provide useful guidance for fishery manager.

4. Compare results for v3 (from exercise 3) with a modified design using five release and return periods. Extend the u and v vectors by using the values from periods 1-2 for the final two periods. How do credible intervals for v3 change when the study is extended?

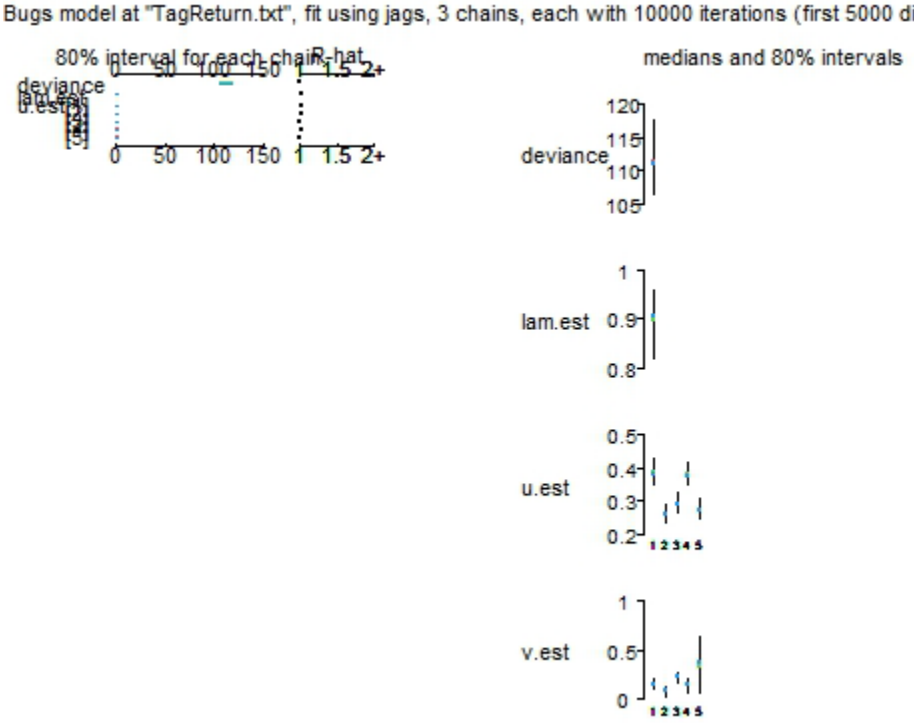
Code changes are limited to the first few lines:

```
# Parameter values for three-yr experiment
Periods <- 5  # Equal number of release and return periods assumed
NTags <- 1000 # Number released each period
NumTagged <- rep(NTags, Periods) # Release NTags fish each period

u.true <- c(0.4, 0.25, 0.3, 0.4, 0.25) # Probability of fishing death
v.true <- c(0.1, 0.15, 0.2, 0.1, 0.15) # Probability of natural death
```

Estimates for v3, using three versus five periods and 1000 tags:

| v3 three | v3 five |
|---|---|
| .02-.69 | .11-.25 |
| .02-.69 | .10-.25 |
| .02-.69 | .15-.29 |

Dramatic reduction in credible interval when study extended (uncertainty shifts to v[5]; see pasted figure below).




5. For the telemetry study, compare results (particularly uncertainty) for parameter v4 when p5 takes on the following values: 0.2, 0.4, 0.7, 0.9. How might this affect planning for conducting the field work?

Code unmodified except for changing p5.

| p5=0.2 | 0.4 | 0.7 | 0.9 |
|---|---|---|---|
| .03-.80 | .02-.63 | .01-.44 | .01-.41 |
| .04-.81 | .03-.74 | .01-.39 | .01-.34 |
| .06-.84 | .02-.58 | .01-.48 | .00-.26 |

Consistent improvement in precision as p5 increases. Could plan for extra searching effort in final period, or possibly add an additional search at end of period 5 (need modified code for additional search).

## Chapter 8

1. Before carrying out simulation runs, describe the expected outcome in fitting the von Bertalanffy growth curve to age:length data if you changed the growth rate (k) to 0.05. Report the results for asymptotic size estimates from several runs. What other simulation setting would need to change in order to accommodate such a slow growth rate?

The expected outcome would be a poor estimate of asymptotic size. Estimates were 74.8, 107.3, 156.6, 71.0 and 88.9 (mostly but not always overestimated). Changing the maximum age to 30 allows for asymptotic growth to be apparent in the plot and produces reliable estimates of asyptotic size.

2. For the model in Section 8.1.1, compare estimates from the default model (vague uniform prior distributions) to a model using vague lognormal priors for L_inf, growth rate (k), and Var_L, and a vague normal prior for t_0.

Code for lognormal version:

```
# Fitting von Bertalanffy curve to simulated data
rm(list=ls()) # Clear Environment

# Choose von Bertalanffy parameter values and other simulation
settings
L_inf <- 80  # e.g., 80-cm fish
k <- 0.3 # Slope
t_0 <- -0.5 # Age at length 0
MaxAge <- 12
SystSample <- 5  # Number per age group
N.AgeLen <- SystSample * MaxAge
Age <- rep(1:MaxAge, each=SystSample)
Var_L <- 10 # Variance about age-length relationship
Len <- rnorm(n=N.AgeLen,L_inf*(1-exp(-k*(Age-t_0))), sd=sqrt(Var_L))

plot(Age, Len, xlab="Age", ylab="Length (cm)")

# Load necessary library
library(rjags)
library(R2jags)

# JAGS code
sink("GrowthCurve.txt")
cat("
model{

# Priors

 L_inf.est ~ dlnorm(0, 1E-6) #dunif(0, 200)
 k.est ~ dlnorm(0, 1E-6) #dunif(0, 2)
 t0.est ~ dnorm(0, 1E-6) #dunif(-5, 5)
 Var_L.est ~ dlnorm(0, 1E-6) #dunif(0,100)   # Variability in length
at age

# Calculated value
 tau.est <- 1/Var_L.est

# Likelihood
 for (i in 1:N.AgeLen) {
    Len_hat[i] <- L_inf.est*(1-exp(-k.est*(Age[i]-t0.est)))
    Len[i] ~ dnorm(Len_hat[i], tau.est)
 } #i
}
    ",fill=TRUE)
sink()
```

```
# Bundle data
jags.data <- list("N.AgeLen", "Age", "Len")

# Initial values
jags.inits <- function(){list(L_inf.est=rlnorm(n=1, meanlog=0,
sdlog=0.5),
                                #runif(n=1, min=0, max=max(Len)),
                              k.est=rlnorm(n=1, meanlog=0, sdlog=0.5),
                                #runif(n=1, min=0, max=2),
                              t0.est=rnorm(n=1, mean=0, sd=1),
                                #runif(n=1, min=-5, max=5),
                              Var_L.est=rlnorm(n=1, meanlog=0,
sdlog=0.5))}
                            #runif(n=1, min=0, max=100))}

model.file <- 'GrowthCurve.txt'

# Parameters monitored
jags.params <- c("L_inf.est", "k.est", "t0.est", "Var_L.est")

# Call JAGS from R
jagsfit <- jags(data=jags.data, jags.params, inits=jags.inits,
                n.chains = 3, n.thin=1, n.iter = 10000,
                model.file)
print(jagsfit)
plot(jagsfit)
```

There was no obvious difference in reliability of results for the two types of priors:

| Uniform priors | | | | Lognormal/normal priors | | | |
|---|---|---|---|---|---|---|---|
| Linf | V_L | k | t0 | Linf | V_L | k | t0 |
| 80.2 | 12.5 | 0.3 | -0.4 | 79.7 | 10.5 | 0.3 | -0.5 |
| 80.0 | 8.1 | 0.3 | -0.5 | 79.9 | 13.0 | 0.3 | -0.6 |
| 79.8 | 10.7 | 0.3 | -0.4 | 80.1 | 9.7 | 0.3 | -0.7 |
| 79.5 | 12.2 | 0.3 | -0.6 | 79.3 | 8.7 | 0.3 | -0.5 |
| 80.3 | 11.4 | 0.3 | -0.5 | 80.1 | 8.4 | 0.3 | -0.5 |

3. For either the default or lognormal case from Exercise #2, modify the code to add the fitted curve to the age:length scatter plot. As a hint, you can either redo the scatter plot after the plot(jagsfit) line or add a # to turn off the jagsfit plot.

Approach used was to generate expected values within JAGS and add those to the list of retained variables. Required passing in MaxAge as an additional data point.

```
# Fitting von Bertalanffy curve to simulated data
rm(list=ls()) # Clear Environment

# Choose von Bertalanffy parameter values and other simulation
settings
L_inf <- 80  # e.g., 80-cm fish
```

```
k <- 0.3 # Slope
t_0 <- -0.5 # Age at length 0
MaxAge <- 12
SystSample <- 5  # Number per age group
N.AgeLen <- SystSample * MaxAge
Age <- rep(1:MaxAge, each=SystSample)
Var_L <- 10 # Variance about age-length relationship
Len <- rnorm(n=N.AgeLen,L_inf*(1-exp(-k*(Age-t_0))), sd=sqrt(Var_L))

# Load necessary library
library(rjags)
library(R2jags)

# JAGS code
sink("GrowthCurve.txt")
cat("
model{

# Priors

 L_inf.est ~ dlnorm(0, 1E-6) #dunif(0, 200)
 k.est ~ dlnorm(0, 1E-6) #dunif(0, 2)
 t0.est ~ dnorm(0, 1E-6) #dunif(-5, 5)
 Var_L.est ~ dlnorm(0, 1E-6) #dunif(0,100)   # Variability in length
at age

# Calculated value
 tau.est <- 1/Var_L.est

# Likelihood
 for (i in 1:N.AgeLen) {
    Len_hat[i] <- L_inf.est*(1-exp(-k.est*(Age[i]-t0.est)))
    Len[i] ~ dnorm(Len_hat[i], tau.est)
 } #i
 for (a in 1:MaxAge){
  LenAge[a] <- L_inf.est*(1-exp(-k.est*(a-t0.est)))
 } #a
}
    ",fill=TRUE)
sink()

# Bundle data
jags.data <- list("N.AgeLen", "Age", "Len", "MaxAge")

# Initial values
jags.inits <- function(){list(L_inf.est=rlnorm(n=1, meanlog=0,
sdlog=0.5),
                              #runif(n=1, min=0, max=max(Len)),
                            k.est=rlnorm(n=1, meanlog=0, sdlog=0.5),
                              #runif(n=1, min=0, max=2),
                            t0.est=rnorm(n=1, mean=0, sd=1),
                              #runif(n=1, min=-5, max=5),
```

```
                                Var_L.est=rlnorm(n=1, meanlog=0,
sdlog=0.5))}
                               #runif(n=1, min=0, max=100))}

model.file <- 'GrowthCurve.txt'

# Parameters monitored
jags.params <- c("L_inf.est", "k.est", "t0.est", "Var_L.est","LenAge")

# Call JAGS from R
jagsfit <- jags(data=jags.data, jags.params, inits=jags.inits,
                n.chains = 3, n.thin=1, n.iter = 10000,
                model.file)
print(jagsfit)
plot(jagsfit)

par(mar=c(5.1, 4.1, 4.1, 2.1)) # Restore default margins
plot(Age, Len, xlab="Age", ylab="Length (cm)")
points(seq(1:MaxAge), jagsfit$BUGSoutput$mean$LenAge, pch=16,
col="red")
```

4. Before carrying out simulation runs, describe the expected outcome in fitting the von Bertalanffy growth curve to age:length data if you changed the maximum age to 4 (e.g., if the stock was heavily exploited). Report the results for asymptotic size estimates from several runs.

Expected result would be to have very poor estimates of asymptotic size. Results for five trials: 71.1, 110.4, 226.2, 272.8, and 78.5 (typically overestimated but not always).

5. Modify the code for the von Bertalanffy age:length analysis to obtain the Age vector using the rgamma function that provided relative ages for the tagging analysis. Find a pair of values for shape and rate that result in reliable estimates of the growth parameters.

Modified code using lognormal/normal priors. Shape=1 and Rate=0.2 provides a wide age range and produces reliable estimates of the four parameters.

```
# Fitting von Bertalanffy curve to simulated data
rm(list=ls()) # Clear Environment

# Choose von Bertalanffy parameter values and other simulation
settings
L_inf <- 80  # e.g., 80-cm fish
k <- 0.3 # Slope
t_0 <- -0.5 # Age at length 0

# Generate vector for relative age at initial capture (true age + t0)
Shape <- 1 # Parameters for gamma distribution
Rate <- 0.2
N.AgeLen <- 40
Age <- trunc(rgamma(n=N.AgeLen, shape=Shape, rate=Rate)) # Age vector
MaxAge <- max(Age)
```

```
hist(Age, xlab="Age", main="")

Var_L <- 10 # Variance about age-length relationship
Len <- rnorm(n=N.AgeLen,L_inf*(1-exp(-k*(Age-t_0))), sd=sqrt(Var_L))

# Load necessary library
library(rjags)
library(R2jags)

# JAGS code
sink("GrowthCurve.txt")
cat("
model{

# Priors

 L_inf.est ~ dlnorm(0, 1E-6) #dunif(0, 200)
 k.est ~ dlnorm(0, 1E-6) #dunif(0, 2)
 t0.est ~ dnorm(0, 1E-6) #dunif(-5, 5)
 Var_L.est ~ dlnorm(0, 1E-6) #dunif(0,100)   # Variability in length
at age

# Calculated value
 tau.est <- 1/Var_L.est

# Likelihood
 for (i in 1:N.AgeLen) {
    Len_hat[i] <- L_inf.est*(1-exp(-k.est*(Age[i]-t0.est)))
    Len[i] ~ dnorm(Len_hat[i], tau.est)
 } #i
}
    ",fill=TRUE)
sink()

# Bundle data
jags.data <- list("N.AgeLen", "Age", "Len")

# Initial values
jags.inits <- function(){list(L_inf.est=rlnorm(n=1, meanlog=0,
sdlog=0.5),
                              #runif(n=1, min=0, max=max(Len)),
                            k.est=rlnorm(n=1, meanlog=0, sdlog=0.5),
                              #runif(n=1, min=0, max=2),
                            t0.est=rnorm(n=1, mean=0, sd=1),
                              #runif(n=1, min=-5, max=5),
                            Var_L.est=rlnorm(n=1, meanlog=0,
sdlog=0.5))}
                           #runif(n=1, min=0, max=100))}

model.file <- 'GrowthCurve.txt'

# Parameters monitored
```

```
jags.params <- c("L_inf.est", "k.est", "t0.est", "Var_L.est")

# Call JAGS from R
jagsfit <- jags(data=jags.data, jags.params, inits=jags.inits,
                n.chains = 3, n.thin=1, n.iter = 10000,
                model.file)
print(jagsfit)
plot(jagsfit)
```

## Chapter 9

1. Fit the Beverton-Holt model using the following values for the F array: runif(n=Y, min=0, max=0.1), runif(n=Y, min=0, max=1.0), and runif(n=Y, min=1.0, max=1.4). Present results from five trials and provide an assessment of the reliability of estimates for each case.

Code same as in book except for replacing fixed F array with the above function call.

| F 0-0.1 | | F 0-1.0 | | F 1-1.4 | |
|---|---|---|---|---|---|
| alpha | beta | alpha | beta | alpha | beta |
| .01 | .41 | .01 | .04 | .04 | .07 |
| .01 | .61 | .01 | .07 | .01 | .11 |
| .01 | .82 | .02 | .07 | .03 | .08 |
| .01 | .38 | .01 | .12 | .02 | .08 |
| .01 | .53 | .01 | .10 | .03 | .08 |

Estimates of alpha/q are generally good (expected value 0.01). Estimates of beta (expected value 0.1) are poor when F is low (no contrast in spawning stock so relationship is poorly defined). Allowing F to vary widely (0-1) without a temporal pattern or using all high Fs seems to work as well as the original example (fixed Fs increasing over time). All three cases drive spawning stock down, making the stock-recruitment pattern apparent.

2. The expected value for alpha (0.01 for our default simulation settings) is the true underlying value divided by the survey catchability, because we are fitting the model using survey indices rather than estimates of absolute abundance. Compare estimates of alpha for the Beverton-Holt and constant recruitment models, using the approach from Section 9.2. Explain any pattern that you detect.

Code was the same as in the book. Estimates for the Beverton-Holt model were 0.02, 0.02, 0.01, 0.03 and 0.02 (true value 0.01). For the constant recruitment case, estimates were 0.13, 0.17, 0.10, 0.26, 0.15. Estimates are higher in the constant R case because alpha in this case represents mean recruitment (survey recruitment index in this case). The mean for the recruitment index is always going to be lower than the estimated asymptote. For the last run, the mean index was 7.9 compared to the estimated asymptote of 59 (so 1/59<<1/7.9).

## Chapter 10

1. For the two-age model, compare true and estimated total abundance using absolute relative errors abs((y-y.hat)/y)

for five trials using capture probabilities\index{capture probability} of 0.1, 0.2, and 0.3. Would you recommend a field trial using a capture probability\index{capture probability} of 0.1?

Simulation code unmodified except for changing capture probability:

```
rm(list=ls()) # Clear Environment

# Simulation to create population size and field data
Y <- 10

LowS <- 0.5 # Adult survival, arbitrary lower bound
HighS <- 0.9 # Arbitrary upper bound
AnnualS <- runif(n=(Y-1), min=LowS, max=HighS)

MeanR <- 400 # Arbitrary mean annual recruitment
N.r <- array(data=NA, dim=Y)
SD_rec <- 0.25 # Low level of lognormal error in recruitment
N.r <- trunc(MeanR * rlnorm(n=Y, 0, SD_rec))
N.a <- array(data=NA, dim=Y)
N.a[1] <- N.r[1] + rpois(n=1, lambda=MeanR)
# Arbitrary year-1 population size
for (y in 2:Y){
  N.a[y] <- N.r[y] + rbinom(n=1, size=N.a[y-1], prob=AnnualS[y-1])
} #y

# Short-term (closed) two-sample population estimate
CapProb <- 0.3 # Fraction of population sampled
n1 <- rbinom(n=Y, size=N.a, prob=CapProb)
# Caught and marked in first sample
n2 <- rbinom(n=Y, size=N.a, prob=CapProb)
# Caught and examined for marks in second sample
m2 <- rbinom(n=Y, size=n2, prob=n1/N.a) # Marked fish in second sample

# Telemetry information on annual survival
TelRelease <- 20 # Number of telemetry tags to release annually
TelTags <- array(data=NA, dim=c((Y-1),Y))
for (y in 1:(Y-1)){
  TelTags[y,y] <- TelRelease
  for (t in ((y+1):Y)){
    TelTags[y,t] <- rbinom(n=1, size=TelTags[y,(t-1)],
                           prob=AnnualS[t-1])
  }} # y,t

# Load necessary library
library(rjags)
library(R2jags)

# JAGS code ###################################
sink("IPM.txt")
cat("
model {

# Priors

 Mean_lnR.est ~ dnorm(0, 1E-6)  # Recruitment uses ln-scale parameter
```

```
 SD_rec.est ~ dunif(0, 10)
 tau <- pow(SD_rec.est, -2)
 for(y in 1:(Y-1)) {
    N.r.est[y] ~ dlnorm(Mean_lnR.est,tau)
     # Hierarchical structure for recruitment
  } #y

for (y in 1:(Y-1)){
  AnnualS.est[y] ~ dunif(0,1)  # Uninformative prior, adult survival
}#y

# Population projection
 N.a.est[1] ~ dpois((n1[1]+1)*(n2[1]+1)/(m2[1]+1)-1)
  # Year-1 total, Chapman mod in case m2=0
  # Needed to start model because no prior year information
 for(y in 2:Y) {
     N.a.est[y] ~ dpois(N.r.est[y-1] + (AnnualS.est[y-1]*N.a.est[y-
1]))
     # Index shifted by one year for recruitment estimates
     } #y

#Likelihoods
# Capture-recapture estimate for total pop size at start of year
  for (y in 2:Y){  # m2[1] used above to get starting pop size
    m2[y] ~ dbin((n1[y]/N.a.est[y]), n2[y])
    } #y

# Telemetry information on survival rate
for (y in 1:(Y-1)){
  for (t in ((y+1):Y)){
    TelTags[y,t] ~ dbin(AnnualS.est[t-1],TelTags[y,(t-1)])
  }} # y, t

  for (y in 1:Y){
   RelErr[y] <- abs((N.a[y]-N.a.est[y])/N.a[y])
  } #y
  MeanErr <- mean(RelErr[])
}
    ",fill=TRUE)
sink()

# Bundle data
jags.data <- list("Y", "n1", "n2", "m2", "TelTags","N.a")

# Initial values
jags.inits <- function(){ list(N.r.est= rlnorm(n=(Y-1), meanlog=10,
                                              sdlog=1),
                               AnnualS.est=runif(n=(Y-1), min=0,
                                                 max=1))}

model.file <- 'IPM.txt'
```

```
# Parameters monitored
jags.params <- c("AnnualS.est", "N.a.est", "N.r.est", "SD_rec.est",
                 "RelErr", "MeanErr")

# Call JAGS from R
jagsfit <- jags(data=jags.data, jags.params, inits=jags.inits,
                n.chains = 3, n.thin=1, n.iter = 40000,
                model.file)
print(jagsfit)
plot(jagsfit)

# Code for plots comparing true values and model estimates
par(mfrow = c(1, 3))
plot(seq(1:(Y-1)),
     jagsfit$BUGSoutput$mean$AnnualS.est, ylab="Annual Survival",
     xlab="Year", type="b", pch=19,
     ylim=range(AnnualS,jagsfit$BUGSoutput$mean$AnnualS.est))
points(AnnualS, col="blue", type="b", pch=17)

plot(seq(1:(Y-1)),
     jagsfit$BUGSoutput$mean$N.r.est,
     ylab="Annual recruitment (years 2-Y)", xlab="",
     type="b", pch=19,
     ylim=range(N.r,jagsfit$BUGSoutput$mean$N.r.est))
points(N.r[2:Y], col="blue", type="b", pch=17)
par(xpd=TRUE) # Turn off clipping to put legend above plot
legend("top",
       legend = c("Est", "True"), col = c("black", "blue"),
       pch = c(19,17), text.col = "black",  horiz = T ,
       inset = c(0, -0.15))
par(xpd=FALSE)

MR_N.hat <- (n1+1)*(n2+1)/(m2+1) -1 # Chapman modification in case
m2=0
plot(seq(1:Y), jagsfit$BUGSoutput$mean$N.a.est,
     ylab="Pop size (includes new recruits)", xlab="Year",
     type="b", pch=19,
     ylim=c(min(N.a,jagsfit$BUGSoutput$mean$N.a.est,MR_N.hat),
            max(N.a,jagsfit$BUGSoutput$mean$N.a.est,MR_N.hat)))
points(N.a, col="blue", type="b", pch=17) # True adult pop size
points(MR_N.hat, col='red', type="b") # Point estimates
```

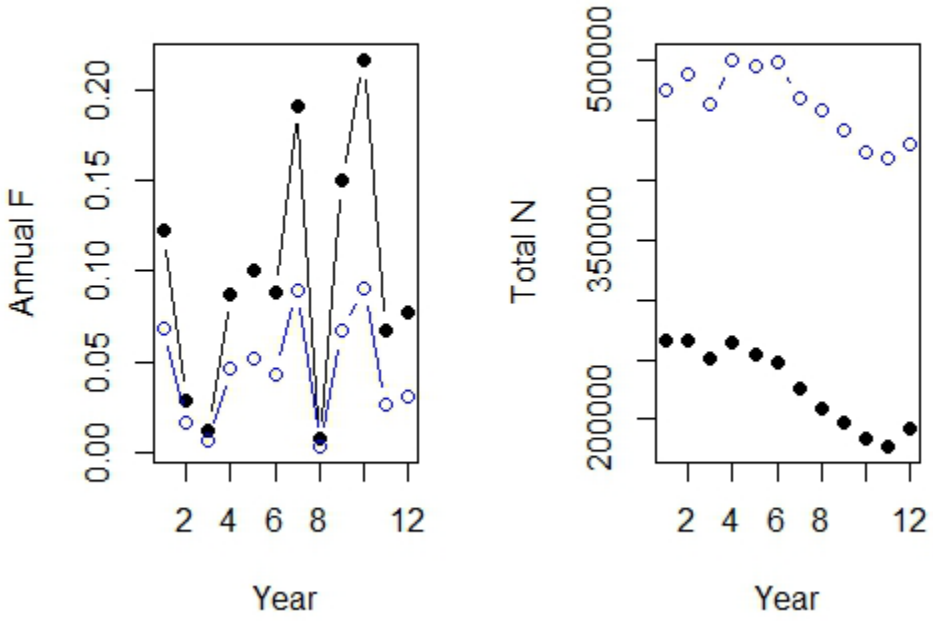
Mean absolute relative errors:

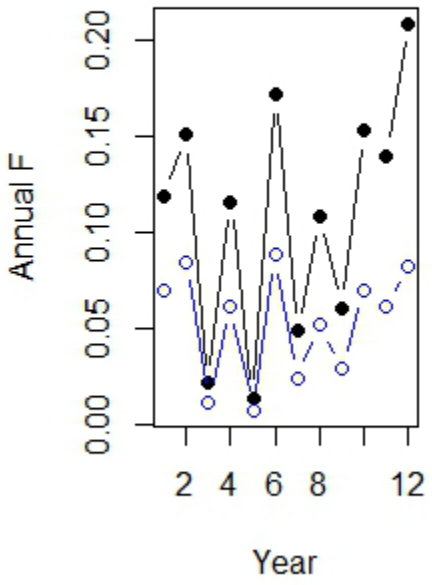
| Capture prob. 0.1 | 0.2 | 0.3 |
|---|---|---|
| 0.61 | 0.11 | 0.11 |
| 9.1 | 0.13 | 0.06 |
| 0.22 | 0.09 | 0.08 |
| 0.17 | 0.13 | 0.08 |
| 0.18 | 0.09 | 0.07 |

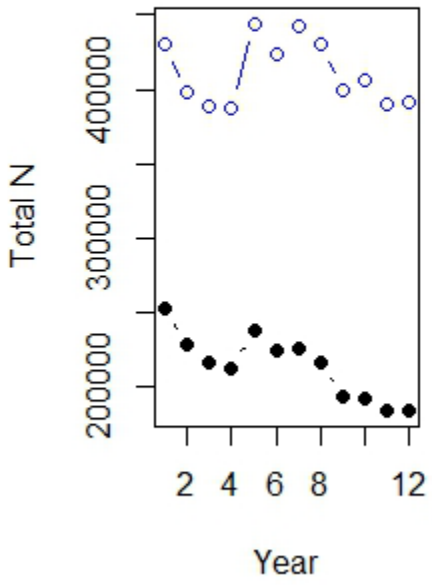
Some runs were acceptable at 0.1 but others had extreme errors so a field study using 0.1 would not be recommended.

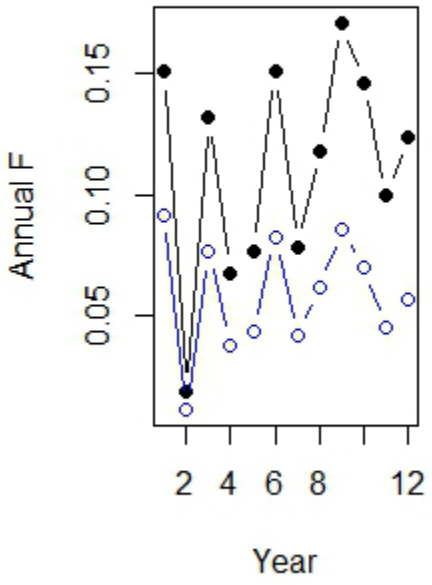
2. For the stock assessment model, set the number of updates at a modest level (e.g., 20,000) and compare the level of convergence when fishing mortality is consistently low (e.g., 0.0-0.1) or high (1.0-1.5). Why is there a difference in how readily convergence is achieved for the two cases?

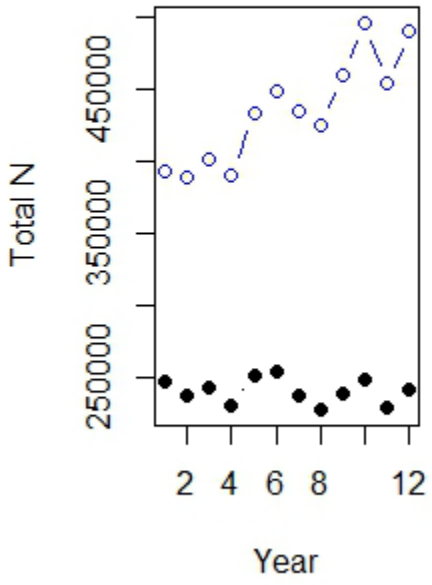
F 0-0.1:



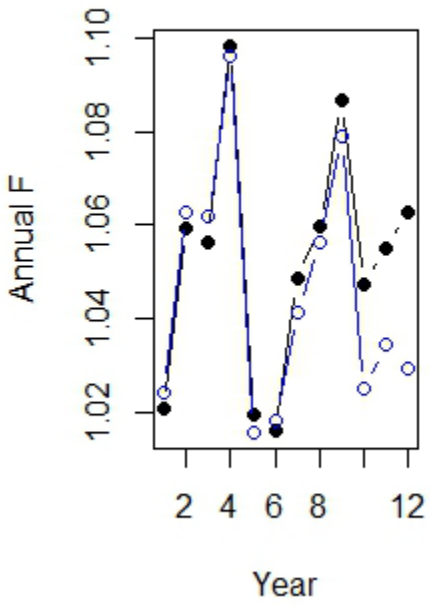

Annual F
0.00
0.05
0.10
0.15
0.20
2
4
6
8
12
Year

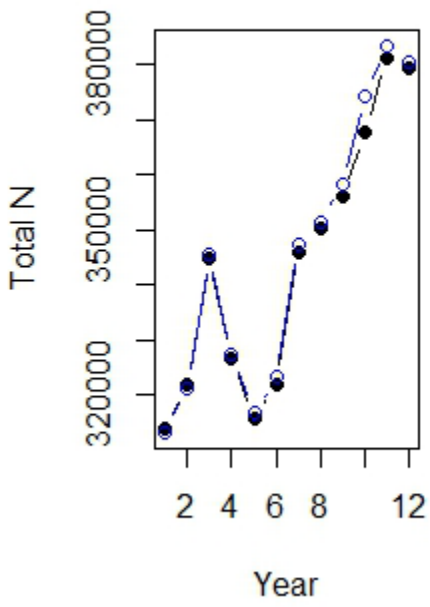

Total N
200000
300000
400000
2
4
6
8
12
Year

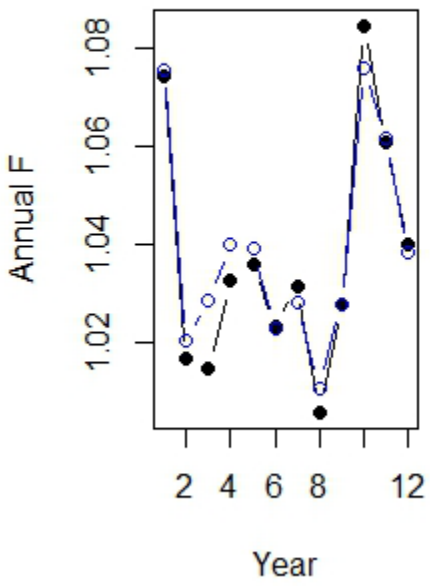

Annual F
0.05
0.10
0.15
2
4
6
8
12
Year

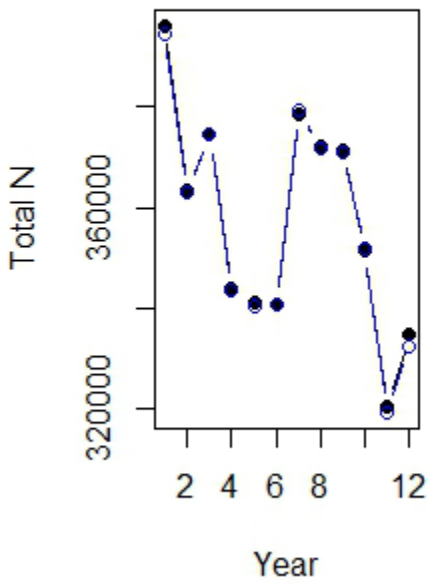

Total N
250000
350000
450000
2
4
6
8
12
Year

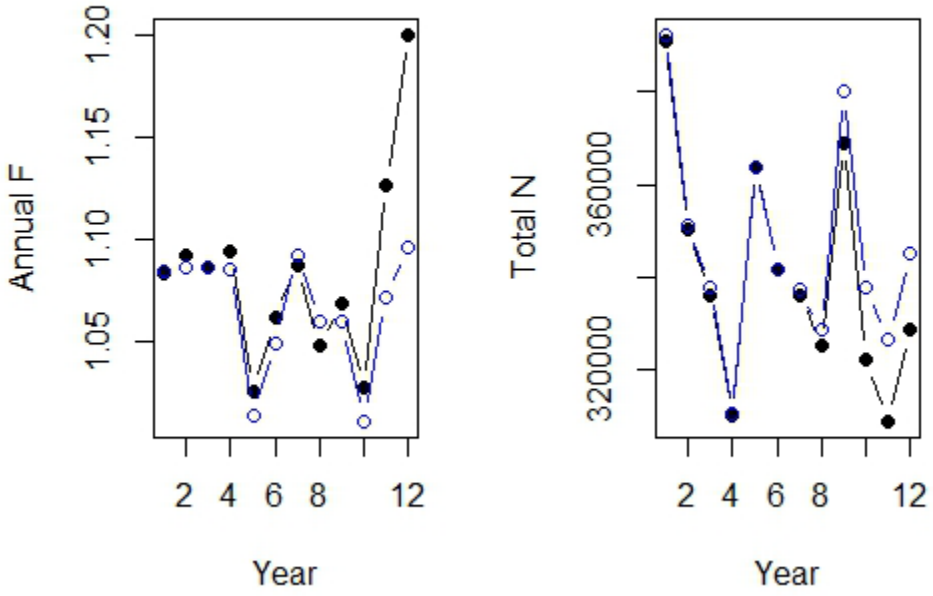
F 1.0-1.5:


Annual F
1.02
1.04
1.06
1.08
1.10
2
4
6
8
12
Year


Total N
320000
350000
380000
2
4
6
8
12
Year


Annual F
1.02
1.04
1.06
1.08
2
4
6
8
12
Year


Total N
320000
360000
2
4
6
8
12
Year




Neither case results in complete convergence in 20,000 updates, but runs are much closer at high F. When F is extremely low, the catch matrix provides much less information about absolute abundance. There is less room for uncertainty about M versus F when F is high, because most fish end up being harvested (=seen).