Mark-recapture method for population size estimation

Week2-ex2, solution

Introduction

This exercise serves also as an example to Bayesian inference and summarizing posterior probability distribution of discrete variables using exact probabilities and Monte Carlo method. Hence, go through the whole file before starting to do the exercise.

Mark-recapture method

The basic idea in mark-recapture method for estimating animal populations is that a researcher visits a study area and captures a group of individuals. Each of these individuals is marked with a unique identifier (e.g., a numbered tag, ring or band), and is released back into the environment. Sufficient time is allowed to pass for the marked individuals to redistribute themselves among the unmarked population. After a while, the researcher returns and captures another sample of individuals.

Assumptions in the basic implementation of the method are, among others, that the time between consecutive captures is long enough for "perfect mixing", marks are not lost, the behavior and capture probability of an individual does not change due to marking and that the study population is "closed". In other words, the two visits to the study area are close enough in time so that no individuals die, are born, move into the study area (immigrate) or move out of the study area (emigrate) between visits. If these assumptions hold, we can reasonably assume that the *marked animals are randomly distributed in the total population" which then allows for inference on the total population size.

This method is illustrated during the lecture where we estimate the number of balls in a bag (the total *population* comprises of all the balls in the bag).

Let N denote the total population size, M the number of marked individuals at first visit, C the total number of animals captured at the second time and R the number of recaptured animals. By assuming that N is large compared to M and that the marked individuals are randomly distributed in the population, we can use Binomial distribution as our observation model for R as follows

$$p(R|C,M,N) = \mathrm{Bin}(R|C,M/N) \tag{1}$$

We have to define a prior for N after which we can solve its posterior

$$p(N|M,C,R) \propto \mathrm{Bin}(R|C,M/N)p(N) \tag{2}$$

The number of marked balls is

```
M=25
```

We will now analyze the total number of balls in the bag. This will be done first by exact calculations with discrete valued N and after that using Markov chain Monte Carlo.

Conduct the inference with discretization

Since there is only one, discrete, variable that we are interested in, we can easily discretize the problem and work with array(s) of probabilities

Let's define an array of values N that we think are a priori plausible at all. The below values are "hard" limits. Prior probability below the minimum and above the maximum is zero

```
abs_min <- M    # the number of balls cannot be negative
abs_max <- 1000  # No way that bag can contain more than 1000 balls (a subjective assumption)


# Define the evaluation points so that all integers between
# abs_min and abs_max are included
Nseq <- seq(abs_min, abs_max, length=abs_max-abs_min+1)
```

Next we define prior for N and draw it.

Now that we have a discrete variable we have to give a prior probability for each of the elements in Nseq. You can do this in multiple ways. Here are few examples:

```
par(mfrow=c(2,3))              # Open figure for plotting the examples

# uniform prior
Nprior <- rep(1,length(Nseq))/length(Nseq)
sum(Nprior)              # check that prior probabilities sum up to to one
```

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

```
plot(Nseq,Nprior, main="Uniform prior", xlab="N", pch=16)
# "Gaussian" prior
Nprior <- dnorm(Nseq, mean=50, sd=20)
Nprior <- Nprior/sum(Nprior)     # Normalize the prior probabilities to sum to one
sum(Nprior)        # check that prior probabilities sum up to to one
```

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

```
plot(Nseq,Nprior, main="Gaussian prior", xlab="N", pch=16)
# log-Gaussian prior
Nprior <- dlnorm(Nseq, mean=5, sd=1)
Nprior <- Nprior/sum(Nprior)   # Normalize the prior probabilities to sum to one
sum(Nprior)        # check that prior probabilities sum up to to one
```

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

```
plot(Nseq,Nprior, main="log-Gaussian prior", xlab="N", pch=16)
# Step wise prior by giving different relative weights for different values
Nprior <- rep(1,length(Nseq))
Nprior[Nseq>50 & Nseq<600] <- 2
Nprior[Nseq>70 & Nseq<400] <- 4
Nprior[Nseq>200 & Nseq<300] <- 6
Nprior <- Nprior/sum(Nprior)    # Normalize the prior probabilities to sum to one
sum(Nprior)        # check that prior probabilities sum up to to one
```

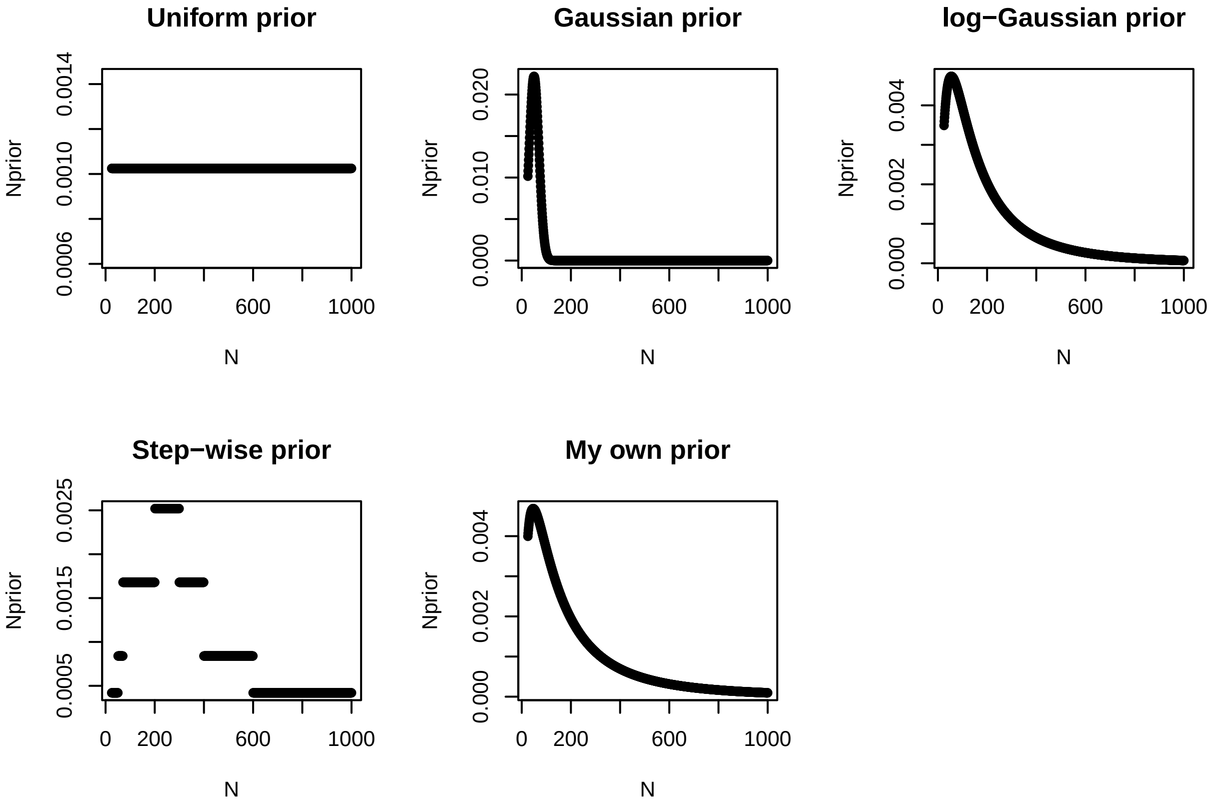
```
## [1] 1
```

```
plot(Nseq,Nprior, main="Step-wise prior", xlab="N", pch=16)

# --- Here we will fill in the prior defined during the lecture ---
Nprior <- dlnorm(Nseq, mean=5.05, sd=1.1)
Nprior <- Nprior/sum(Nprior)    # Normalize to sum to one
sum(Nprior)                     # check that prior probabilities sum up to to one
```

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

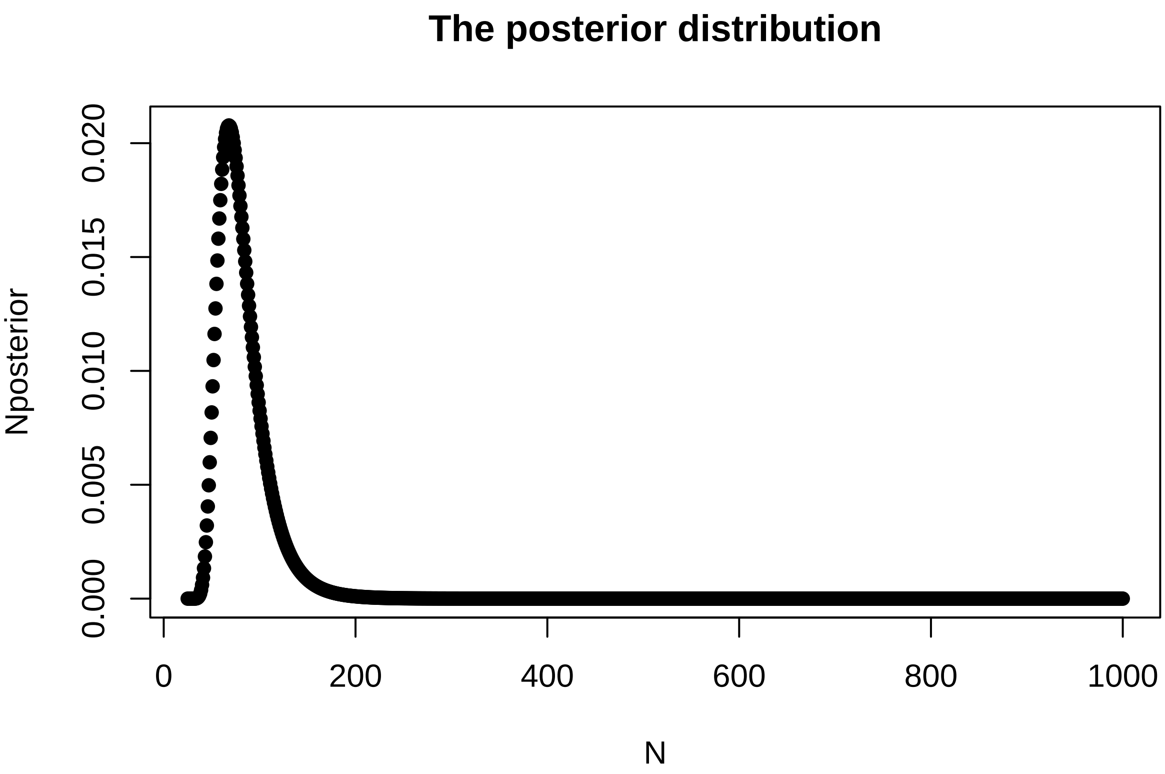
```
plot(Nseq,Nprior, main="My own prior", xlab="N", pch=16)
```



Now that we have defined the vector of prior probabilities for different values of N we can can conduct the second sampling round, to obtain data C and R, and after that calculate the posterior distribution for it by using the Bayes Theorem explicitly

```
# The result from the other sampling time
C=25
R=9

Nposterior <- Nprior*dbinom(R,C,M/Nseq)  # numerator of Bayes theorem
Nposterior <- Nposterior/sum(Nposterior) # divide by marginal likelihood
plot(Nseq,Nposterior, main="The posterior distribution", xlab="N", pch=16)
```



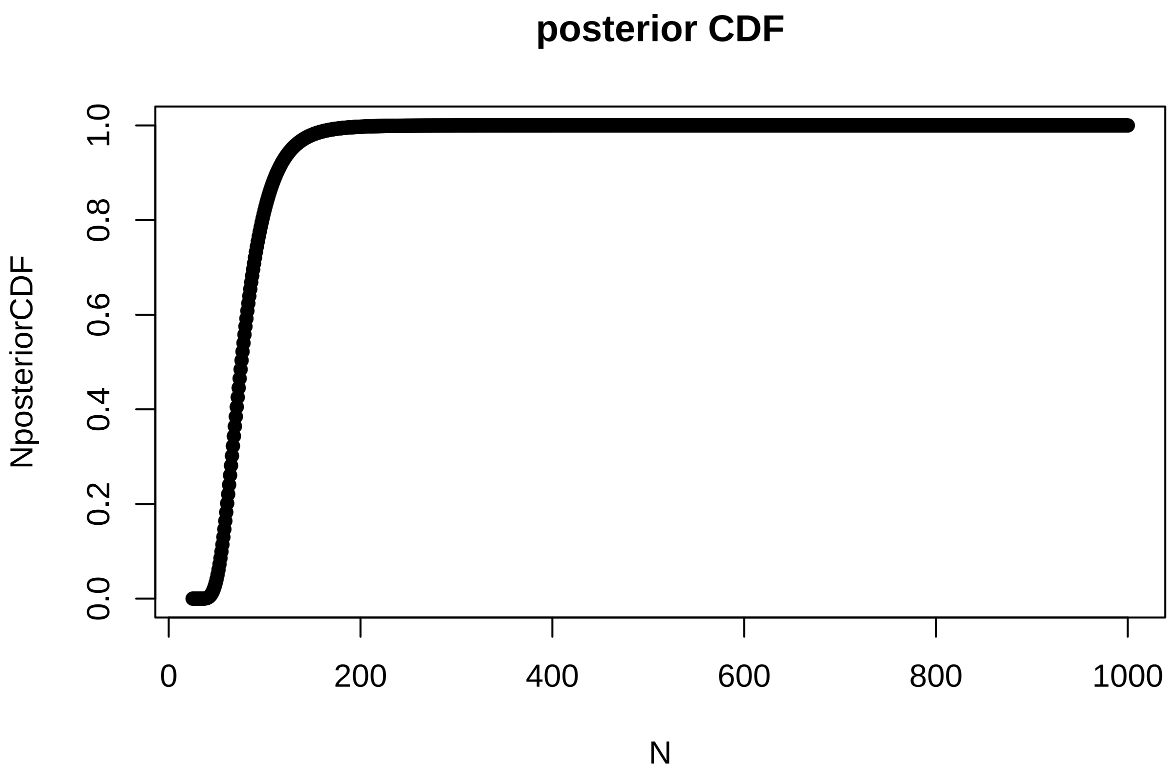
Given the vector of posterior probabilities for different values of N we can calculate various summaries for the posterior distribution. Such as the posterior mean.

```
posteriorMean = sum(Nposterior*Nseq)
print(posteriorMean)
```

```
## [1] 81.44218
```

In order to calculate quantiles, we need to first calculate the cumulative distribution function

```
NposteriorCDF <- cumsum(Nposterior)
# Plot CDF
plot(Nseq,NposteriorCDF, main="posterior CDF", xlab="N", pch=16)
```



Now we can calculate, for example, the 90% posterior quantile

```
# 10% quantile is the first N at which CDF is over or equal to 10%
Nseq[which(NposteriorCDF>=0.1)[1]]
```

```
## [1] 56
```

```
# 90% quantile is the first N at which CDF is over or equal to 90%
Nseq[which(NposteriorCDF>=0.9)[1]]
```

```
## [1] 113
```

Analysis using Monte Carlo

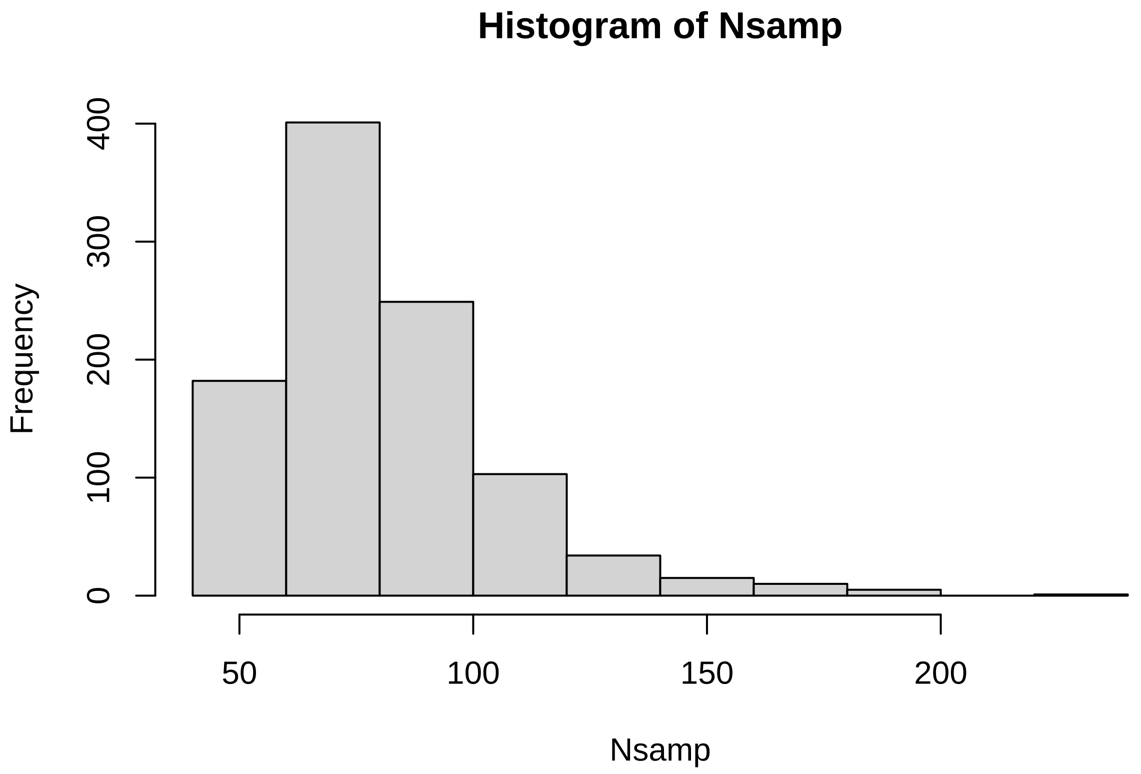
Next, we conduct the analysis using Monte Carlo technique. Here the idea is to draw random samples from the posterior distribution of N and then use these to calculate summaries of the posterior distribution.

In this example, we will draw the samples using a technique that utilizes the inverse cumulative distribution function of the posterior. Note, this technique is not part of the learning goals of the course so you can jump over the next code block. However, if you are interested in the technique see page 23 in BDA3.

```
Nsamp = Nseq[sapply(runif(1000,min=0,max=1),function(temp){ which(NposteriorCDF>=temp)[1] })]
```

We can now use the vector Nsamp as a collection of random samples from the posterior distribution. Hence, let's draw the histogram of them and calculate the mean and 10 % and 90 % quantiles

```
hist(Nsamp)
```



```
mean(Nsamp)
```

```
## [1] 80.988
```

```
quantile(Nsamp,probs=c(0.1,0.9))
```

```
##   10%   90%
##  55.0 112.1
```

Exercise for week 2

Calculate the posterior median, variance, standard deviation, and 80% central posterior interval using:

1. the vectorized form of the posterior probablity distribution (that is, the vectors Nseq, Nposterior, NposteriorCDF)
2. the Monte Carlo approximation using the random samples from the posterior that are stored in Nsamp
3. Additionally, why aren't the results from the above two approaches identical? How could you get them to match in theory?

Model solution

1 vectorized form of the posterior distribution

```
# posterior median
posteriorMedian = Nseq[which(NposteriorCDF>=0.5)[1]-1]
print(posteriorMedian)
```

```
## [1] 75
```

```r
# Posterior variance
posteriorVariance = sum(Nposterior*(Nseq-posteriorMean)^2)
print(posteriorVariance)
```

```
## [1] 637.6833
```

```r
# posterior standard deviation
posteriorSd = sqrt(posteriorVariance)
print(posteriorSd)
```

```
## [1] 25.25239
```

```r
# 80% central posterior interval
interval80 = cbind(Nseq[which(NposteriorCDF>=0.1)[1]] , Nseq[which(NposteriorCDF>=0.9)[1]])
print(interval80)
```

```
##      [,1] [,2]
## [1,]   56  113
```

2 Monte Carlo approach

```r
# posterior median
posteriorMedianMC=median(Nsamp)
print(posteriorMedianMC)
```

```
## [1] 76
```

```r
# Posterior variance
posteriorVarianceMC = var(Nsamp)
print(posteriorVarianceMC)
```

```
## [1] 623.7957
```

```r
# posterior standard deviation
posteriorSdMC = sqrt(posteriorVarianceMC)
print(posteriorSdMC)
```

```
## [1] 24.9759
```

```r
# 80% central posterior interval
interval80MC = quantile(Nsamp,c(0.1,0.9))
print(interval80MC)
```

```
##   10%   90%
##  55.0 112.1
```

3 Difference between the approaches

```r
# medians
cbind(posteriorMedian,posteriorMedianMC)
```

```
##      posteriorMedian posteriorMedianMC
## [1,]              75                76
```

```r
# variances
cbind(posteriorVariance,posteriorVarianceMC)
```

```
##      posteriorVariance posteriorVarianceMC
## [1,]          637.6833            623.7957
```

```
# standard deviations
cbind(posteriorSd,posteriorSdMC)
```

```
##      posteriorSd posteriorSdMC
## [1,]    25.25239       24.9759
```

```
# 80% central posterior intervals
rbind(interval80,interval80MC)
```

```
##              10%   90%
##               56 113.0
## interval80MC  55 112.1
```

Clearly each of the summary statistics is similar with Monte Carlo and the discretized approach so that the actual numbers are close to each others. However, since we use only 1000 samples in Monte Carlo there is random variation in it. In theory we would get the true result with Monte Carlo if we increased the sample size to infinity.

Grading

Total points 10: Four points from question 1 so that 1 point from correct implementation and result from each of the four summary statistics. Similarly, four points from question 2. Two points from, question 3 so that correct answer to each of the two sub-questions gives 1 point.