R Demonstration – Analysis of Averages

Objective: The estimation of averages is an important step in many ecological analyses. In this demonstration you will be introduced to R and OpenBUGS functions which will allow you to reach this goal.

Part I. Plotting histograms and calculation using a frequentist approach

For this lesson, you will need to download three files from the course website to the `PCB6466` folder on your Desktop: 1) the text file containing the *Hypericum cumulicola* height data (`Hcdata.txt`); 2) the text file containing the *Hypericum cumulicola* population means height data (`popmeanHc.txt`), and 3) the file containing the R script (`Analysis of Averages.R`). You will also need to install the add-on package R2OpenBUGS to communicate with OpenBUGS. In the program below, the variables and settings of the model are specified in the R script and sent to OpenBUGS, so the analysis is run directly in R.

Open the R script `Analysis of Averages.R`. To see how this script works, let's take a look at the code. For this part we will concentrate in the upper portion of the script.

```
rm(list=ls())
getwd()
# example of a directory
setwd("C:/Users/Pedro/Documents/Classes/Methods in Ecology/2012 fall")
# Before running the script make sure you change the working directory to yours

Hc_data <- read.table("Hcdata.txt", header=T)
Hc_pop  <- read.table("popmeanHc.txt", header=T)

### Analysis of complete dataset ###
Height_data_95 <-Hc_data$h95[Hc_data$SA95==1 & Hc_data$h95<90]
hist(Height_data_95,100,main="Histogram of Hypericum cumulicola height (cm)")
abline(v=Hc_pop$pop_mean, col="blue")
abline(v=mean(Height_data_95), col="red")
hist(Hc_pop$pop_sd^2)
mean(Height_data_95)
round(sqrt(var(Height_data_95))/sqrt(length(Height_data_95)),4)
summary(lm(Height_data_95~1))
```

The first line in the program `rm(list=ls())`allows you to clear the memory. This is a good practice when you start a new program. The second and third lines are used to obtain the current directory and set the directory that will be used in the program. You will need to modify this line to adapt it to your directory pathway or you can do it manually from the File menu. The function `read.table("file name.txt", header=T)` obtains the data.

Since 1994, Quintana-Ascencio *et al.* have collected demographic data of *Hypericum cumulicola* in 14 populations at Archbold Biological Station, Florida USA (Quintana-Ascencio 2003). Here we use the data for 1995. The line:

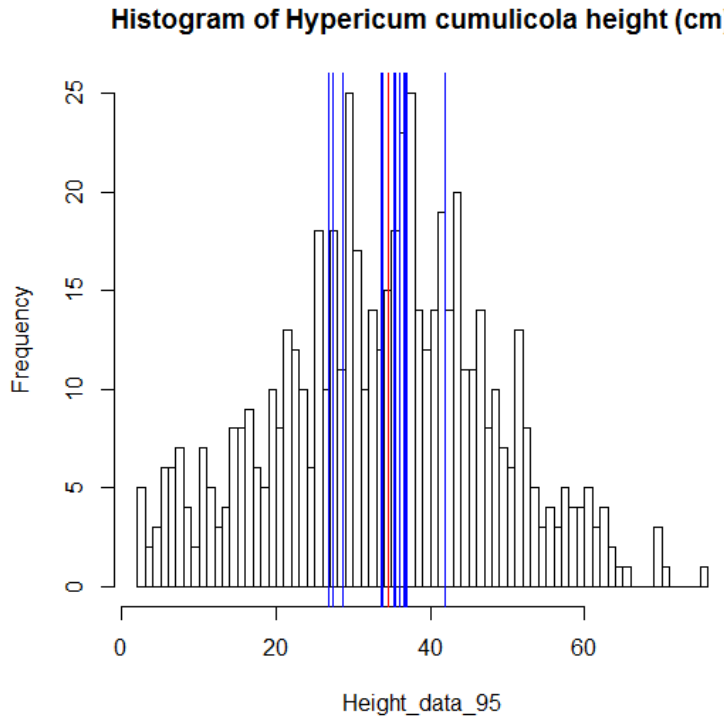
```
Height_data_95 <-Hc_data$h95[Hc_data$SA95==1 & Hc_data$h95<90]
```

filters the plant height data for 1995 to include only plants older than one year (SA95==1) and eliminates a questionable data point (because of its extreme size) we detected in a prior demonstration (`h95<90`). The same line allocates the filtered data to a new variable `Height_data_95.`

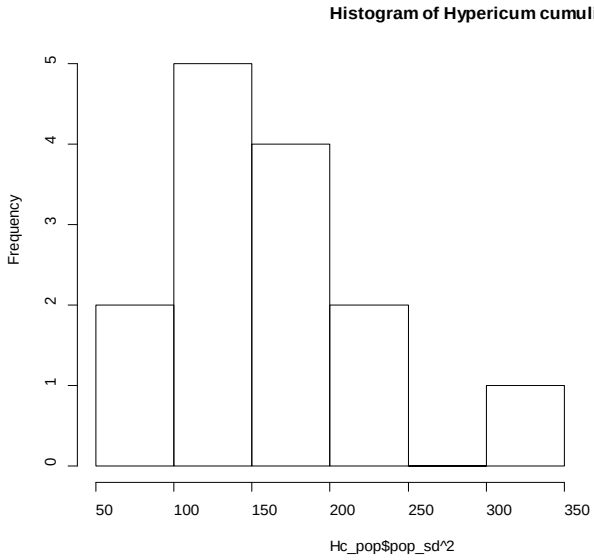
It is always useful to check the distribution of the data. We will use the function `hist()` to create a histogram of adult plant heights, and the function `abline()`to add in blue the location of the 14 population means and in red the location of the overall mean.

```
hist(Height_data_95,100,main="Histogram of Hypericum cumulicola
height (cm)")
abline(v=Hc_pop$pop_mean, col="blue")
abline(v=mean(Height_data_95), col="red")
```

Your plot should look like the one below where the data seems to follow a normal distribution:



We also plot a histogram of the population variances that tend to follow a log normal distribution.



There are alternative ways to obtain an average in R. Here we first use the function `mean(Height_data_95)` and get an overall plant height mean of 34.47 cm. We also obtain the value of the standard error of the plant height mean as 0.5801 cm with a series of R functions. We calculate the square root of the variance and divide it by the square root of the sample size and round the result to four digits.

```
round(sqrt(var(Height_data_95))/sqrt(length(Height_data_95)),4)
```

R has functions that can accomplish these calculations in fewer steps.

```
summary(lm(Height_data_95~1))
```

The output is as follows:

```
Call:
lm(formula = Height_data_95 ~ 1)
Residuals:
   Min     1Q  Median     3Q    Max
-32.469  -8.969   0.531   9.531  41.531
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  34.4689     0.5801   59.42   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 14.15 on 594 degrees of freedom
```

Part II. Calculation of averages using a Bayesian approach

We will now describe the second part of the code:

```
##Sampling from the dataset###

size <-10
sample_height <- sample(Height_data_95,size)
n <- length(sample_height)
x <- sample_height

pop.mean.mean <- mean(Hc_pop$pop_mean)
pop.mean.var <- var(Hc_pop$pop_mean)
log.pop.var.mean <- mean(log(Hc_pop$pop_sd^2))
log.var.var <- var(log(Hc_pop$pop_sd^2))

## Call  the packages

library(R2OpenBUGS)
library(coda)

### Bayesian Analysis with Uninformed priors###
# Write model
meanmodel<-function()
{
## Priors
  mean_height ~ dnorm(0,1.0E-6)  #uninformative prior of the mean height of the plants
  var_height ~ dlnorm(0.0,1.0E-6)  #uninformative prior of the variance of the height of the plants
  prec <- 1/var_height

## Likelihood
for (i in 1:nobs) # for each plant
  {
  Y[i] ~ dnorm(mean_height, prec)  # height from a normal distribution
  }
}

write.model(meanmodel,"meanmodel.txt")
```

```
# Bundle data
mean.data <- list(Y=x, nobs=n)

# Inits function
inits <- function()
  list(var_height=100,mean_height=100)

# Parameters to estimate
params <- c("mean_height","var_height")

# MCMC settings
nc = 1
ni=100000
nb=1000
nt=100

# Start Gibbs sampler
output_model <- bugs(data = mean.data, inits = inits, parameters = params, model =
"meanmodel.txt",
n.thin = nt, n.chains = nc, n.burnin = nb, n.iter = ni, codaPkg=T)

line.coda<-read.bugs(output_model)
summary(line.coda)
```

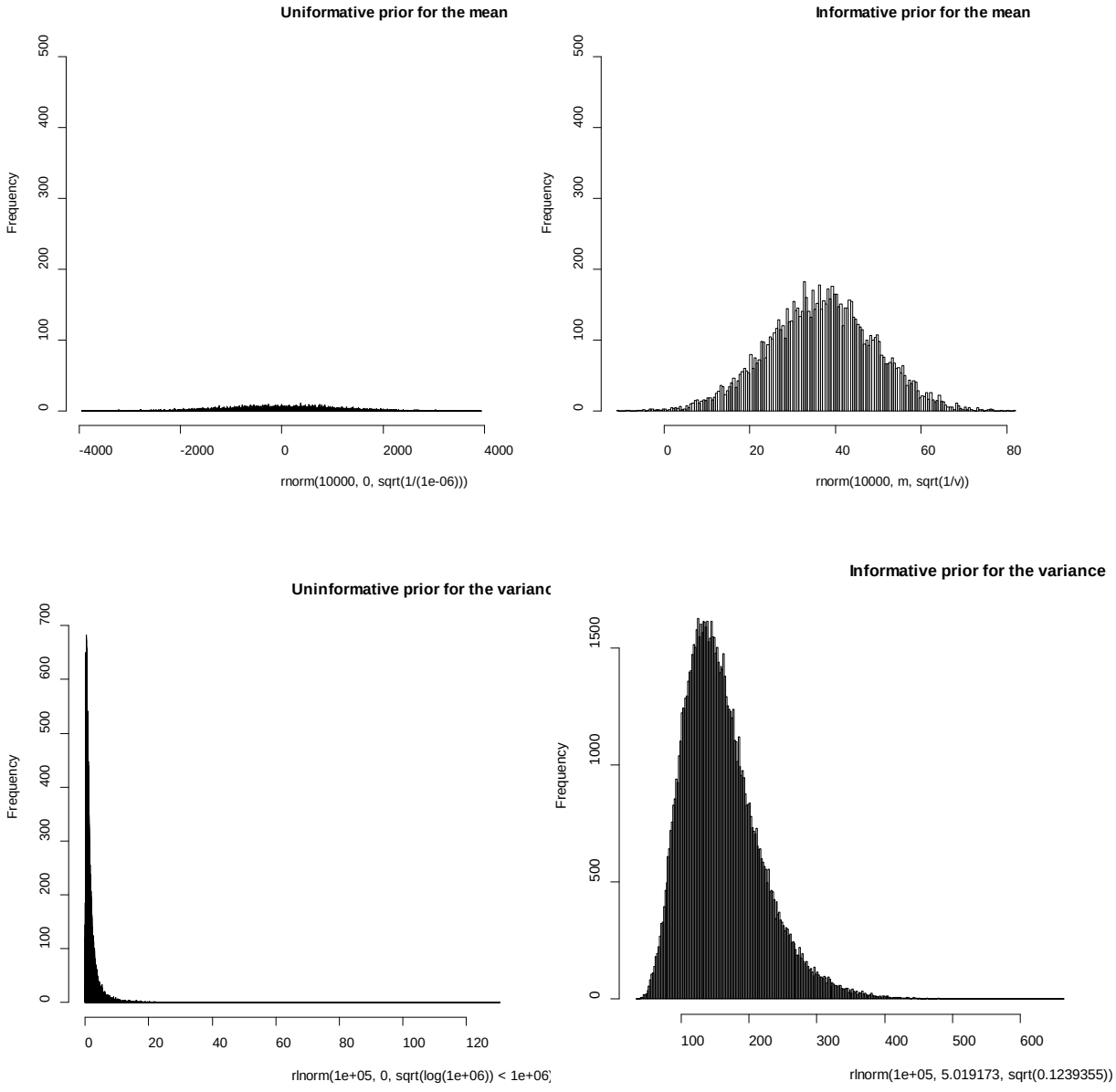
We start by obtaining a sample of 10 plant heights. We allocate ten into size and use for this purpose the function `sample()`

```
size <- 10
sample_height <- sample(Height_data_95,size).
```

Next, we create several variables that will be used in the model: `n` is the sample size, `x` is a vector that contains that data, `pop.mean.mean` is the overall population mean, `pop.mean.var` is the population variance, `log.pop.var.mean` is the mean of the logarithm of the variances and `log.var.var` is the variance of the logarithm of the variances.

We use the functions `library(R2OpenBUGS)`and `library(coda)` to call the program that connects R with OpenBUGS, and write the OpenBUGS program as described above(*). We define two priors. The first one is an uninformative prior for the mean `dnorm(0,1.0E-6)`, the second one is an uninformative prior for the variance `dlnorm(0.0,1.0E-6)`. Notice that the first prior describes a normal distribution while the second corresponds to a lognormal distribution. Consider that in OpenBUGS functions the variance (1,000,000) is indicated as precision (1/variance). This

uninformative prior for the mean has a mean of zero and a large variance (1,000,000) implying our assumed lack of prior information. Later we will use an informed prior based on the data for one of the populations for the mean `dnorm(36.59, 0.006514144)`and the overall information for the variance `dlnorm(5.019, 0.00651)`. Compare their distributions in the plots below. Notice that the specification of the distributions is not consistent between R and OpenBUGS.



We now need to specify the calculation of the likelihood for each element of the sample. For more information read McCarthy (2007).

We allocate the data to `mean.data`, assign the initial values for the Markov chain, specify the parameters that will be obtained, and entered the settings for the Monte Carlo Markov chain

(MCMC): nc = number of chains, ni = number of iterations, nb = number of chains discarded, nt = the frequency for extracting values.

The function `bugs()`determines the procedure for the model and allocates the results into the variable `output_model`. A realization of the program is presented below.

```
Iterations = 1001:1e+05 Thinning interval = 1 Number of chains = 1
Sample size per chain = 99000
1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:
              Mean     SD Naive SE Time-series SE
deviance     75.61  2.204 0.007005       0.007005
mean_height  29.59  3.603 0.011451       0.011451
var_height  129.07 81.135 0.257863       0.257863
2. Quantiles for each variable:
              2.5%   25%    50%    75%  97.5%
deviance    73.46 74.04  74.94  76.48  81.51
mean_height 22.42 27.35  29.59  31.83  36.72
var_height  47.71 79.54 108.20 152.70 336.80
```

We obtain an estimate of the sample mean and its 95% credible interval that it is commensurable to a realization obtained from a frequentist approach obtained with the following functions (remember that the results will vary for each sample):

```
### Frequentist Analysis ###

Freq_results <- array(0,c(1,4))
colnames(Freq_results) <- c("mean","sd", "2.5", "97.5")
rownames(Freq_results) <- "mean_height"
Freq_results[1,1] <- sample_mean <- mean(sample_height)
Freq_results[1,2] <-sample_std_error <- sd(sample_height)/sqrt(size)
## calculate the upper and lower limits of the 95% confidence interval
Freq_results[1,3] <- sample_mean - qt(0.975, size-1)*sample_std_error
Freq_results[1,4] <- sample_mean + qt(0.975, size-1)*sample_std_error
Freq_results
```

```
            mean   2.5%    97.5%
mean_height 29.6 22.4369 36.7631
```

However in this case we have prior relevant information. We have the mean from each population and the variance among populations. You can think of a scenario where two different people sample independently the *Hypericum cumulicola* population. The first person collects a more extensive census of several populations while the second one takes a smaller overall sample of 10 individuals. The second person uses as prior the information from the larger sample. We use as priors the plant

height mean and standard deviation among populations (34.47, 17.54), and the mean and variance of the logarithm of the variance among populations (5.02, 0.124).

The data available for the second person is the sample of 10 individuals but she incorporates the informed priors for the estimation of the mean and variance distributions. We need to replace the following lines of code in the R script part to calculate informed parameters of the prior (be careful when locating them in the script):

```
pop.mean <- pop.mean.mean
pop.var  <- 1/pop.mean.var
mean.data <- list(Y=x, nobs=n, m = pop.mean, v=pop.var,
mv=log.pop.var.mean, vv=log.var.var)
```

and two more lines in the OpenBUGS model to allocate them in the model:

```
  mean_height ~ dnorm(m,v)   #informative prior of mean
  var_height ~ dlnorm(mv,vv) #informative prior of variance
```

A realization of the program is presented below.

```
Iterations = 1001:1e+05 Thinning interval = 1 Number of chains = 1
Sample size per chain = 99000
1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:
              Mean     SD Naive SE Time-series SE
deviance     75.45  1.948 0.006190       0.006190
mean_height  31.52  2.678 0.008512       0.008512
var_height  126.56 74.149 0.235660       0.235660
2. Quantiles for each variable:
             2.5%   25%    50%    75%  97.5%
deviance    73.46 74.03  74.88  76.27  80.60
mean_height 26.43 29.72  31.44  33.24  37.02
var_height  48.19 79.80 107.80 150.70 315.80
```

This realization has a much narrower credible interval (26.43-37.02) for the same sample than those estimated with the frequentist approach (22.43-36.76) or the uninformed Bayesian model (22.42-36.72). Remember the actual values will change because of the randomness of the sampling process.

Part III. Exercise # 3

1. Change the code to evaluate a larger sample of 100 plants. Calculate the frequentist estimate and the uninformed and informed Bayesian estimates. Report the estimated mean and standard deviation, and the confidence interval and credible intervals of the estimated means.
2. Sample the data from the populations (n=5 and n=3) and use the information from these sampled populations as an informed prior. Report the estimated mean and standard deviation, and the confidence interval and credible intervals of the estimated means.

3. Calculate the *mean*, *sum of squares*, *variance*, and *standard deviation* of the height for a sample of 10 *Hypericum cumulicola* in 1995. DO NOT use the shortcut functions (e.g., `mean`, `sd`, etc.) that R provides, but instead use the formulas in presentations. Present the data and the calculations in Excel.
4. Another fundamental theorem of probability is the Central Limit Theorem. According to page 65 of Gotelli & Ellison (2004): "the Central Limit Theorem (Chapter 2) shows that arithmetic means of large samples of random variables conform to a normal or Gaussian distribution, *even if the underlying random variable does not* [emphasis added]."

   Download and run the R script `Central_Limit_Theorem.R` from the course website. Try changing some of the variables in the script, such as the number of samples or the sample size. Also, try to use a different probability distribution than the Poisson to demonstrate the Central Limit Theorem. Did you get the same result?

References

Quintana-Ascencio, P. F., E. S. Menges, and C. Weekley. 2003. A fire-explicit population viability analysis of *Hypericum cumulicola* in Florida rosemary scrub. *Conservation Biology* **17**: 433-449.

McCarthy. M.A. 2007. Bayesian Methods for Ecology. Cambridge University Press.

* **How to Install a package in R...**

1. In the menu Packages go to Install package(s)...
2. Select a CRAN mirror to download the package from (it makes absolutely no difference which one you choose).
3. A window with a list of available packages sorted alphabetically should appear.
4. Scroll until you find the package you need (e.g. coda).
5. It may ask you if you want to use a personal library (at least in BL 305), say Yes and then say Yes again to creating this library in the Documents folder (in your personal computers you may store this wherever you want).
6. The package should download and install itself automatically, you can check that it worked by calling the package e.g. library(coda), ignore the warning message of different versions, usually this causes no problems.