

## R Demonstration – Summary Statistics and the Law of Large Numbers

**Objective:** The purpose of this session is to use some of the R functionality you have recently learned to demonstrate the Law of Large Numbers. Then, you will be introduced to additional R functions, which contain some more advanced programming logic.

### Part I. Demonstrating the Law of Large Numbers in R

The **Law of Large Numbers** states that as the size of a sample drawn from a random variable increases, the mean of more samples gets closer and closer to the true population mean  $\mu$ . This fundamental theorem of probability is fairly straightforward to demonstrate in R (though, as you will soon discover, the process can be quite tedious).

First, start the R software. Next, let us assume that we somehow know that the height of *Hypericum cumulicola* adult plants is normally distributed with a mean of 34.5 cm and a standard deviation of 14.15 cm. We assign these values to the variables  $\mu$  and  $\sigma$  as follows (remember, we use Greek letters for population parameters):

```
> mu <- 34.5  
> sigma <- 14.15
```

Next, we will create a vector named `mean_vector` to hold our sample means:

```
> mean_vector <- rep(0, 30)
```

The `rep` function, as used above, creates a vector of length 30 that initially contains all zero values. Every time we calculate a sample mean, we will store it in this vector.

Now you will proceed to create 30 random samples from our normal distribution, and these samples will increase in size from  $n=1$  to  $n=30$ . For the first sample, type:

```
> sample <- rnorm(1, mean=mu, sd=sigma)  
> mean_vector[1] <- mean(sample)
```

The first line uses the `rnorm` function to draw a sample of size 1 from our normal distribution with the `mu` and `sigma` parameters we specified earlier. The second line uses the `mean` function to calculate the arithmetic mean of the sample and then stores it in the first index of the `mean_vector` variable.

To fill in the rest of the vector, we will increase the sample size by one each time. To save some typing, however, we will combine the process of drawing the random sample, calculating the sample mean, and storing the value in `mean_vector` into a single step. Thus, for a sample size of 2, type the following:

```
> mean_vector[2] <- mean(rnorm(2, mean=mu, sd=sigma))
```

The `rnorm` function will draw a sample of size 2 from our hypothetical distribution, the `mean` function will then calculate the sample mean, and then the mean will get stored in the second index

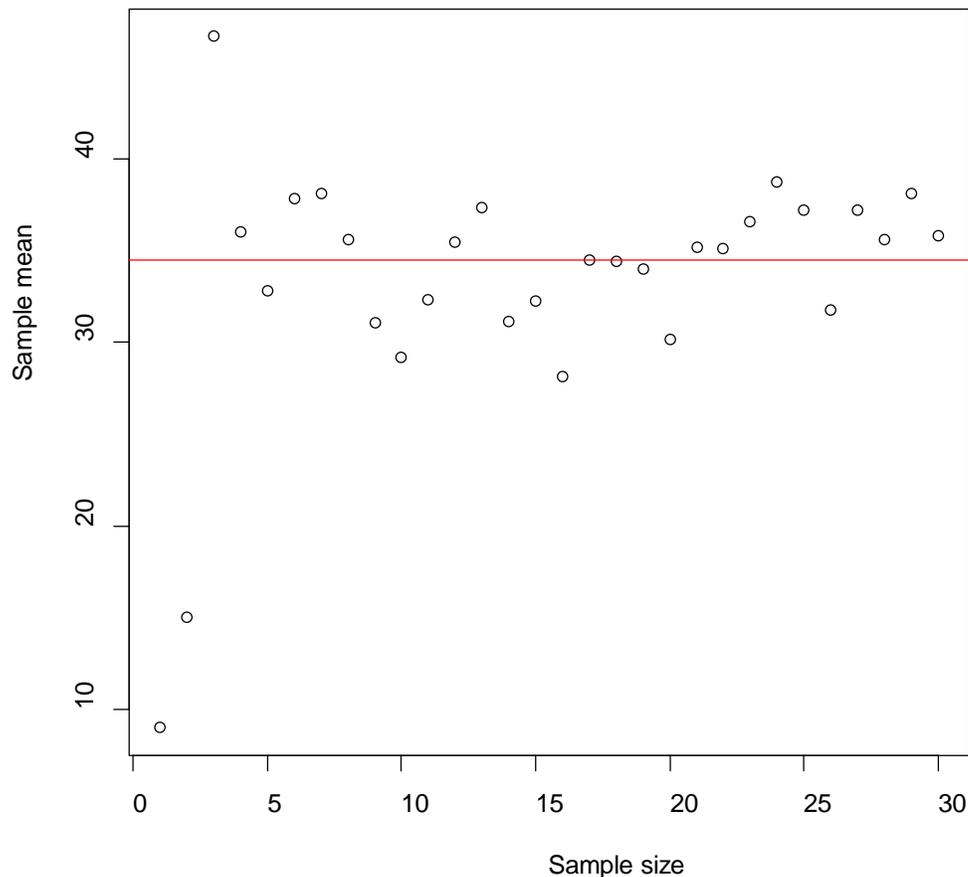
of the `mean_vector` variable. Now for the tedious part: you will need to repeat this process for all of the remaining sample sizes. Use the Up-Arrow key and then change the index position and the sample size, as illustrated below:

```
> mean_vector[3] <- mean(rnorm(3, mean=mu, sd=sigma))  
...  
...  
> mean_vector[30] <- mean(rnorm(30, mean=mu, sd=sigma))
```

Finally, we will create a scatterplot of the results using the `plot` function:

```
> plot(seq(1,30), mean_vector)  
> abline(h=mu, col="red")
```

The `seq(1,30)` argument to the `plot` function generates a sequence of integers from 1 to 30 on the X-axis for all of the sample sizes we computed, and the `abline` function on the next line draws a red horizontal line showing the value of our population mean (i.e.,  $\mu=34.5$ ). While your specific results will be different, your scatterplot should look similar to the following:



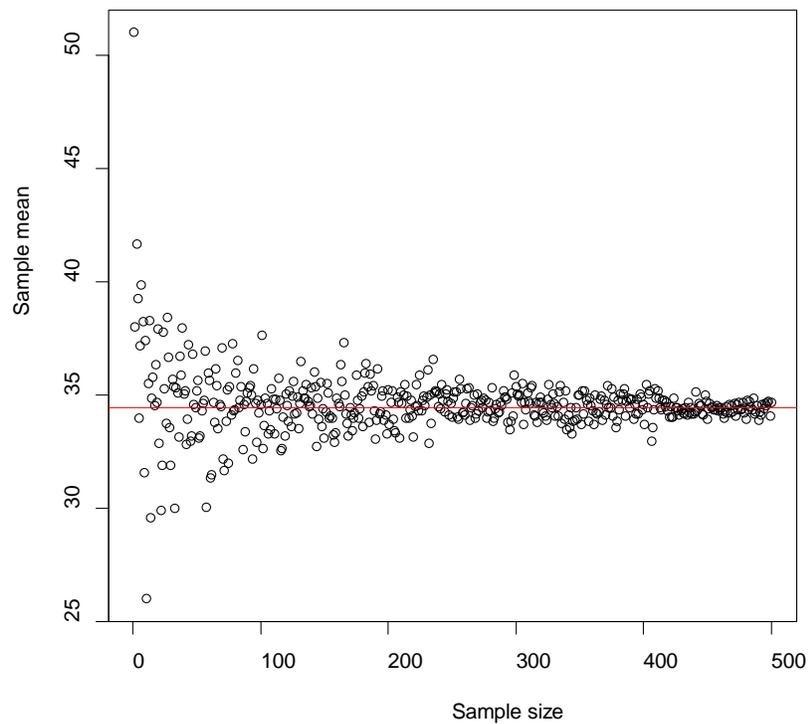
As predicted by the Law of Large Numbers, when our sample size increases from 1 to 30, more estimated means get closer and closer to the population mean. To really see the Law of Large Numbers in action, however, we need to dramatically increase the sample size. And, since we don't want to have to compute all of those samples by hand, we will see how to write a simple R script to automatically do this for us.

## **Part II. Your First R Script**

For this part of the lesson, you will need to download a couple of files from the course website to the PCB6466 folder on your Desktop: 1) the Excel spreadsheet containing the *Hypericum cumulicola* height data (Hcdata.xls); we need to paid attention to the data for 1995 when 595 plants were measured, and 2) the file containing the R script (Law\_of\_Large\_Nums.R).

Next, open the Excel spreadsheet and navigate to the **data** tab. Using the procedure you learned in the lesson before, save the data as a tab-delimited text file named Hc\_data.txt. NOTE: you should open the text file and make sure that there is a column heading for the data named h95, otherwise the R script won't run properly. Also remember that we need to filter the wrong outlier.

If you closed R, start it again. Choose File→Change dir... from the menu bar and then select your PCB6466 folder. Next, choose File→Open script... from the menu bar and then choose Law\_of\_Large\_Nums.R from the list of files. This will open the script in a new window inside of R. Finally, choose Edit→Run all from the menu bar. When the script finishes running, you should see a similar output:



As promised earlier, this script provides an even more dramatic illustration of the Law of Large Numbers. You can clearly see the sample means (the dots) converge on the population mean (denoted by the red line) as the sample size increases from 1 to 500.

To see how the **Law\_of\_Large\_Nums.R** script works, let's take a look at the code:

```
## the following variable holds the maximum sample size
max_sample_size <- 500

## this vector will hold the mean calculated for each sample size
mean_vec <- rep(0, max_sample_size)

## Illustrate the law of large numbers by calculating means for all
## of the different samples from size n=1 to n=max_sample_size.

for (n in 1:max_sample_size) {

  ## draw a random sample of length n from the plant height data
  values <- sample(h95[Hc_data$SA95==1 & Hc_data$h95<90], n)
  ## notice that we are filtering the data to only include adult plants and remove the outlier

  ## calculate the sample mean and store it in the mean_vec
  mean_vec[n] <- mean(values)
}

## finally, plot the sample mean vs. the sample size
plot(seq(1,max_sample_size), mean_vec, xlab="Sample size", ylab="Sample mean")

## draw a red horizontal line with a Y-intercept equal to mu
abline(h=mu, col="red")
## Do not forget to detach the data
detach(Hc_data)
```

As shown above, a script is just a collection of R commands stored in a text file. To execute the commands, you can 1) choose **Edit**→**Run all** from the menu bar to run the entire script in a line-by-line fashion, or 2) select one or more lines of text in the script and then choose **Edit**→**Run line or selection** from the menu bar, or 3) copy one or more lines of code from the script and paste directly into the R Console.

An important thing to note is that **any line that begins with the # character will not be executed**. These lines are known as “comments” in programming lingo, and they are used to explain to other readers what the various parts of our script do. While only a single # character is required to denote

a comment line, we use double # characters (as in the script above) because we think they stand out more.

Now let's take a line-by-line look at the script to see what each part does. The first line is a comment, and the following 2 lines read the *Hypericum cumulicola* height data from the text file into a data frame named `Hc_data` and then execute the `attach` function to make the variable `h95` visible in the R session. Remember, you can see what variables are contained in the data frame by typing `names(Hc_data)` at the R Console after you have executed `attach`.

The lines above create 3 new variables. The first is `mu`, which holds the height mean computed from our population of 595 *Hypericum cumulicola* adult plants. The second is `max_sample_size`, which indicates that our maximum sample size will be 500. And the third is `mean_vec`, which is a vector that will hold the sample means we will calculate. We use the `rep` function to initialize this vector with a sequence of zero values.

Now we get to the portion of the script that may seem unfamiliar to you:

```
## Illustrate the law of large numbers by calculating means for all
## of the different samples from size n=1 to n=max_sample_size.
for (n in 1:max_sample_size) {
  ## draw a random sample of length n from the plant height data
  values <- sample(h95[Hc_data$SA95==1 & Hc_data$h95<90], n)

  ## notice that we are filtering the data to only include adult plants and remove the
  outlier

  ## calculate the sample mean and store it in the mean_vec
  mean_vec[n] <- mean(values) }
```

First notice that we are filtering the data to only include adult plants and remove the typo we detected in the prior demonstration. After the 2 commented lines, you see a line in bold that begins with the R command **for**. In programming lingo, the block of code from this line until the bold **}** (7 lines below) is known as a “for loop”. The way it works is this: when R gets to the `for` command, it evaluates the expression in parentheses (`n in 1:max_sample_size`). This expression assigns the value 1 to the variable `n`, which is known as the counter variable for the for loop. Then R executes the block of code between the `{}`. The first line of code uses the `sample` function to draw a random sample of size `n` from our data, and the second line computes the sample mean and stores it in the first position of our `mean_vec` vector.

The important concept to grasp here is that, unlike the previous commands we've looked at, R will not simply move on to the next lines after executing these two lines. Rather, it will go back into the (`n in 1:max_sample_size`) expression and increase the value of `n` by one. It will then execute again the lines of code between the `{}`, only this time with the new value of `n`. R will keep doing this until `n` exceeds the value of the variable `max_sample_size`, at which time execution will continue below the for loop.

In a nutshell, then, the `for` loop shown above counts from a sample size of `n=1` to `n=max_sample_size`. Each time, it draws a sample of size `n` from the data, computes the sample mean, and stores it in the `mean_vec` vector. (NOTE: if you are confused about how the `for` command works, consult Crawley (2005) or the R help system).

Once we've used our `for` loop to fill in all the sample mean values in `mean_vec`, we simply use the `plot` function to create a graph of sample size versus sample mean:

**Congratulations—you've just become an R programmer!**