R: Introduction

To begin, download R from the R-Project web site ([www.r-project.org](http://www.r-project.org)). R is different from most statistical packages in that it contains a very primitive interface (though this is continually improving) and as a result has a more hands-on or programming feel than other statistical software packages.

The standard installation of R utilizes a default user interface. In this course, we will instead use RStudio which provides a richer interface. Once you have installed the R base package, download RStudio from the following website: <http://www.rstudio.com/>.

R is an open source package. This has both advantages and disadvantages. Because it is open source, there is no “software support” you can directly access; however, there are literally *thousands* of documents on the web that can help you use R efficiently (some are much better than others). The following links give some of the most popular webpages for R support.

* <http://cran.r-project.org/other-docs.html>
* <http://cran.r-project.org/manuals.html>
* <http://cran.r-project.org/doc/manuals/R-intro.html>

R’s base package contains many basic functions used in statistics. In addition to the base package, many individuals have created other packages that can be downloaded that will aid in various analyses.

The following provides a snapshot of the R Studio interface that is commonly used when using R.



Organizational Structure

The organization structure in R is best managed through what is called Projects. To create a new Project, select File > New Project …



|  |  |
| --- | --- |
| First, specify New Directory  | Next, to begin we will create an Empty Project |

Specify the name and location for the new directory that will contain this new project.

|  |  |
| --- | --- |
| Next, specify the name and location fo this new directory | Verify that the new direcory and project has been created |

The frame in the upper-left is your script window and the frame on the lower-left is the R console window. You can enter command directly into the R console; however, I’d encourage you to get accustomed to using the script window. The following can be used to obtain an R script window.



**Getting Started**

To get started, lets create a vector named x1. This can be done by typing the following command at the > prompt.

> x1 <- c(1,2,3,4)

To view the contents of the vector, simple specify its name and hit Enter.

> x1

[1] 1 2 3 4

Simple calculations on this vector can easily be carried out. For example, you can add 1 to all elements of the vector as follows:

> x1+1

[1] 2 3 4 5

Other mathematical operators can be used, as well. For example, try x1\*2, x1^2, and sqrt(x1).

Summary statistics are also easily obtained using some basic functions that exist in the basic package of R. To calculate the mean of x1, simply type the following.

> mean(x1)

[1] 2.5

You can also compute the variance.

> var(x1)

[1] 1.666667

Next, try to compute the standard deviation as shown below.

> stdev(x1)

Error: could not find function "stdev"

Why doesn’t this work? R recognizes that the standard deviation function would be redundant and thus it is not included in the base package. To obtain the standard deviation of x1, simply type the following.

> sqrt(var(x1))

[1] 1.290994

**Reading in Data Files**

To open an existing data file in RStudio, select **Import Dataset** in the window shown in the upper-right. Choose to import data from a Text File.



Choose to read in the **Skull.txt** file, and the following window should appear:



Click Import, and the data set will be added to your workspace. If you click on the data set name in your workspace, the data set will appear in the upper-left window.



R stores data in what are known as data.frames. You can think of these as matrices; however, R technically treats them differently.

You can see the variable names by typing the command **names()** at the prompt.

> names(Skull)

[1] "TimePeriod" "MaxBreadth" "BaseHeight" "BaseLength" "NasalHeight"

You can see the dimension of the data.frame in the following window.



This data.frame is shown below.



You can refer to each element in this data.frame in a way that is similar to how elements of a matrix are identified in R. For example, **Skull[1,1]** will return the value in the 1st row and 1st column of the data.frame.

> Skull[1,1]

[1] 4000BC

Similarly, the value in the 1st row, 3rd column can be obtained.

> Skull[1,3]

[1] 138

The entire first row can be displayed by leaving the column position empty.

> Skull[1,]

 TimePeriod MaxBreadth BaseHeight BaseLength NasalHeight

1 4000BC 131 138 89 49

The first three rows can be displayed with the following command:

> Skull[1:3,]

 TimePeriod MaxBreadth BaseHeight BaseLength NasalHeight

1 4000BC 131 138 89 49

2 4000BC 125 131 92 48

3 4000BC 131 132 99 50

To see the entire set of MaxBreadth values, enter the following.

> Skull[,2]

 [1] 131 125 131 119 136 138 139 125 131 134 129 134 126 132 141 131 135 132 139

[20] 132 126 135 134 128 130 138 128 127 131 124 124 133 138 148 126 135 132 133

[39] 131 133 133 131 131 138 130 131 138 123 130 134 137 126 135 129 134 131 132

[58] 130 135 130 137 129 132 130 134 140 138 136 136 126 137 137 136 137 129 135

[77] 129 134 138 136 132 133 138 130 136 134 136 133 138 138

You can easily obtain the mean for the MaxBreadth variable.

> mean(Skull[,2])

[1] 132.7333

**Summarizing Data in R**

The format of a data frame is akin to the table structure in Excel.

|  |  |  |
| --- | --- | --- |
|  | Excel | R |
| Structure Name | Table | Data.frame |
| Referencing a field | Skull[MaxBreath] | Skull$MaxBreath |

The following command returns an error because the data frame has not been referenced.

> mean(MaxBreadth)

Error in mean(MaxBreadth) : object 'MaxBreadth' not found

Instead, we can easily obtain the mean of MaxBreadth.

> mean(Skull$MaxBreadth)

[1] 132.7333

To get the average of all the remaining variables, you can enter the following set of commands in the R Script window. Once you have written the commands, highlight them and select **Run**.

> mean(Skull$MaxBreadth)

> mean(Skull$BaseHeight)

> mean(Skull$BaseLength)

> mean(Skull$NasalHeight)

The following appears in your Console:

> mean(Skull$BaseHeight)

[1] 133.3667

> mean(Skull$BaseLength)

[1] 98.08889

> mean(Skull$NasalHeight)

[1] 50.44444

This code could be made more efficient using the **apply()** function in R. The following is a snippet of the documentation obtained by entering **help(apply)** at the command.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Usage**apply(X, MARGIN, FUN, ...)**Arguments**

|  |  |
| --- | --- |
| X |  the array to be used. |
| MARGIN |  a vector giving the subscripts which the function will be applied over. 1  indicates rows, 2 indicates columns, c(1,2) indicates rows and columns. |
| FUN |  the function to be applied: see ‘Details’. In the case of functions like +, %\*%,  etc., the function name must be backquoted or quoted. |
| ... |  optional arguments to FUN. |

 |

To get the mean for each *numerical* variable in this data set, you could use the following command:

> apply(Skull[,2:5],2,mean)

 MaxBreadth BaseHeight BaseLength NasalHeight

 132.73333 133.36667 98.08889 50.44444

Suppose you also wanted the variance for each numerical variable in the data set. You could use the **apply()** function as follows.

> apply(Skull[,2:5],2,var)

 MaxBreadth BaseHeight BaseLength NasalHeight

 21.748315 21.852809 26.329089 9.463171

Next, try to find the standard deviation as follows:

> apply(Skull[,2:5],2,stdev)

What happens? Find a way to calculate the standard deviation for each numerical variable in R.

Finally, note that the **summary** function can also be used in the **apply()** function.

> apply(Skull[,2:5],2,summary)

 MaxBreadth BaseHeight BaseLength NasalHeight

Min. 119.0 121.0 87.00 44.00

1st Qu. 130.0 130.2 94.25 48.00

Median 133.0 134.0 98.00 50.00

Mean 132.7 133.4 98.09 50.44

3rd Qu. 136.0 136.0 101.00 53.00

Max. 148.0 145.0 114.00 60.00

Notice that the first argument in the **apply()** command used above contains only the columns for which a mean can be computed. The following command will not work and produces this error.

> apply(Skull,2,mean)

 TimePeriod MaxBreadth BaseHeight BaseLength NasalHeight

 NA NA NA NA NA

Warning messages:

1: In mean.default(newX[, i], ...) :

 argument is not numeric or logical: returning NA

2: In mean.default(newX[, i], ...) :

 argument is not numeric or logical: returning NA

3: In mean.default(newX[, i], ...) :

 argument is not numeric or logical: returning NA

4: In mean.default(newX[, i], ...) :

 argument is not numeric or logical: returning NA

5: In mean.default(newX[, i], ...) :

 argument is not numeric or logical: returning NA

Likewise, the following command does not work because there is no ‘margin’ to apply as Skull[,2] is a single vector and does not contain multiple columns.

> apply(Skull[,2],2,mean)

Error in apply(Skull[, 2], 2, mean) : dim(X) must have a positive length

To summarize categorical variables, you should use the **table()** function. For example, the following command returns the number of observations in each time period.

> table(TimePeriod)

TimePeriod

1850BC 3350BC 4000BC

 30 30 30

To obtain the percentages instead of the counts, enter the following:

> table(TimePeriod)/length(TimePeriod)

TimePeriod

 1850BC 3350BC 4000BC

0.3333333 0.3333333 0.3333333

You can also multiply each percentage by 100:

> table(TimePeriod)/length(TimePeriod)\*100

TimePeriod

 1850BC 3350BC 4000BC

33.33333 33.33333 33.33333

Above, we obtained the summaries for each numerical variable, but this was across all time periods; here, we’d like the summaries of each of these variables for *each* time period. That is, our goal is to obtain the mean for each variable **BY** each time period.

First, let’s look at the help file for the **by()** function.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Usageby(data, INDICES, FUN, ..., simplify = TRUE)Arguments

|  |  |
| --- | --- |
| data | an **R** object, normally a data frame, possibly a matrix. |
| INDICES | a factor or a list of factors, each of length nrow(data). |
| FUN | a function to be applied to data frame subsets of data. |
| ... | further arguments to FUN. |
| simplify | logical: see [tapply](http://127.0.0.1:19244/library/base/help/tapply). |
| Examplesattach(warpbreaks)by(warpbreaks[, 1:2], tension, summary)by(warpbreaks[, 1], list(wool = wool, tension = tension), summary)by(warpbreaks, tension, function(x) lm(breaks ~ wool, data = x)) |

 |

Enter the following command, and R returns the summaries by Time Period.

> by(Skull[,2:5], TimePeriod, summary)

TimePeriod: 1850BC

 MaxBreadth BaseHeight BaseLength NasalHeight

 Min. :126.0 Min. :123.0 Min. : 87.00 Min. :45.00

 1st Qu.:132.2 1st Qu.:131.0 1st Qu.: 92.25 1st Qu.:48.25

 Median :136.0 Median :133.5 Median : 96.00 Median :50.00

 Mean :134.5 Mean :133.8 Mean : 96.03 Mean :50.57

 3rd Qu.:137.0 3rd Qu.:137.0 3rd Qu.: 99.75 3rd Qu.:52.75

 Max. :140.0 Max. :145.0 Max. :106.00 Max. :60.00

------------------------------------------------------------

TimePeriod: 3350BC

 MaxBreadth BaseHeight BaseLength NasalHeight

 Min. :123.0 Min. :124.0 Min. : 90.00 Min. :45.00

 1st Qu.:130.0 1st Qu.:129.2 1st Qu.: 97.00 1st Qu.:48.00

 Median :132.0 Median :133.0 Median : 98.50 Median :50.50

 Mean :132.4 Mean :132.7 Mean : 99.07 Mean :50.23

 3rd Qu.:134.8 3rd Qu.:136.0 3rd Qu.:101.75 3rd Qu.:52.75

 Max. :148.0 Max. :145.0 Max. :107.00 Max. :56.00

------------------------------------------------------------

TimePeriod: 4000BC

 MaxBreadth BaseHeight BaseLength NasalHeight

 Min. :119.0 Min. :121.0 Min. : 89.00 Min. :44.00

 1st Qu.:128.0 1st Qu.:131.2 1st Qu.: 95.00 1st Qu.:49.00

 Median :131.0 Median :134.0 Median :100.00 Median :50.00

 Mean :131.4 Mean :133.6 Mean : 99.17 Mean :50.53

 3rd Qu.:134.8 3rd Qu.:136.0 3rd Qu.:102.75 3rd Qu.:53.00

 Max. :141.0 Max. :143.0 Max. :114.00 Max. :56.00