Basic R Tutorial

Importing and Saving a Data Set

Read In A CSV:

Type Into R: getwd()

This will return the directory R is looking at. Place the CSV file into this directory and then type

read.csv(file="filename.csv")

Importing a Data Set

Command - varname<-read.table(file.choose( ),header=TRUE,sep=",")

Varname is the abbreviation for "variable name" or the title of the attribute. Name the variable so that it is easily recognizable and not easily forgotten! After executing the command, R will open up a dialog box which will allow you to browse for documents saved on the computer. Find the data set you want to import and click open.

Saving Data Sets in Excel

When saving data sets in Excel, select the CSV MS-DOS from the Save As Drop down menu. Then click "yes," to retain the Excel format.

Commands

R syntax is case sensitive. Therefore, the name of the categories you use in R must match the name of the categories in the actual data set itself. The same syntax applies to commands, and object names. For example, if you wanted to calculate median then the command has to be "median," not MEDIAN or Median or any other variant.

Declaring and Initializing Variables

First choose the name of the variable you would like to use and then to initialize to a specific category of data, first use the variable name used to import the data and then $categoryName.

Command x=trial$Age

In this example x is the variable name that I am creating, trial is the variable name of the whole data set and Age is the specific category name I wanted to assign the data to. *Remember R is case sensitive so remember to use the exact data set names as found in the excel document.

Now that you have declared and initialized the Age category to the variable x, you can use it in all of your calculations instead of typing in varname$categoryName. To display the data the variable hold simply enter the variable name and press enter.

Summary

Command: summary(varname)-Insert the actual name of the variable in the parentheses, which will then provide the "summary" of the data set selected. Note the "summary" command uses the same variable name as the variable title within the original data set. The summary command will return the following: minimum, 1st quartile, median, mean, 3rd quartile and maximum.

Median

Commands: median(varname) – To display the median of all the different categories of data

median(varname$Age)- To display the median of a specific category of data such as age

Mean

Commands: mean(varname) – To display the mean or average of all the different variables of the data set

mean(varname$Age) - To display the mean of a specific category of data such as age

Standard Deviation

Commands: sd(varname) – To display the standard deviation or distance measure of all The different variables of data

sd(varname$Age)- To display the mean of a specific category of data such as age

Plot

Commands: plot(varname) – To display the graph of all The different variables of data

plot(varname$Age) – To display the graph of the category of data such as Age.

IQR

Commands: IQR(varname) – To display the IQR or distance measure of all The different variables of data

IQR(varname$Age) – To display the IQR of the category of data such as Age.

Logarithm

Commands: log (varname) – To display the Log or distance measure of all The different variables of data

log(varname$Age) – To display the log of the category of data such as Age

Histogram

Commands hist (varname) – To display the histogram or distance measure of all The different variables of data

hist(varname$Age) – To display the Histogram of the category of data such as Age

Single T-Test

Command t.test(varname)-To display the single t.test or distance measure of all The different variables of data
t.test(varname$Age)-To display the single t.test or distance measure of all The different variables of data such as Age

Independent 2-group T-Test

Command t.test(varname)-To display the Independent 2-group t-test or distance measure of all The different variables of data

t.test(varname$Age)-To display the Independent 2-group t-test or distance measure of all The different variables of data such as Age

Confidence interval

Commands:

First find the mean of the variable in this case

mean(varname$Age)

---

then assign it to a using the syntax

a <- whatever you got for the mean

Then assign the standard deviation of the variable to s using the same syntax

s <- sd(varname$Age)

then put the sample size in the variable n in this case n <- 10. Then put in the equation

error <- qnorm(0.975)*s/sqrt(n)

do the left side with the syntax

left <- a-error

do the right side with the syntax

right <- a+error

and then get your answers by typing simply

right

and

left

Scatter Plot

Command	x=varname$Age
	y=varname$Education
	Declare and initialize two quantitative variables you want to compare.*See declaring and initializing variable for reference* To create the table(x-y coordinates) use the command head(cbind(x-axis variable name, y-axis

variable name)). To plot the scatter plot use the command plot(x-axis variable name, y-axis variable name

+xlab=”Name of the x-axis”

+ylab=”Name of the y-axis”)

Sample execuction of the code

x=trial$Age

y=trial$Education

head(cbind(x,y)) →To create a table of the data points of x-y coordinates

plot(x,y,

+ xlab="Age",

+ ylab="Education") → To graph the scatter plot with the given Data

Loess Smoothing Curves

Command

Import the data set. Declare and Initialize variables for the x-y coordinates and then plot the data set using the plot(x,y) command. Then apply the smoothing line and use the default value of 0.8 for span. If needed you can modify the span value. After applying the loess smoothing line compute all the smoothing vales along the curve and then plot the curve.

Sample Code Execution

age=varname$Age → Declaring and initializing variables

education=varname$Education

plot(age,education)→ Plotting the graph using the declaration above

y.loess <- loess(y ~ x, span=0.8, data.frame(x=age,y=education)) → smoothing curve/ span default value =0.8

y.predict <- predict(y.loess, data.frame(x=age) → computing the values

lines(age,y.predict) → plotting the curve.

title(main="Loess Smoothing Curves")

CrossTabulation:

First create a table with the variables needed , where varname$Age and varname$Education are your specified. i.e. varname$Changethistoyourvariablename

mytable <- table(varname$Age, varname$Education)

Then to generate the frequencies of the first variable summed over the second do:

margin.table(mytable,1)

To get the frequencies of the second variable summed over the first do:

margin.table(mytable,2)

To get the cell percentages do:

prop.table(mytable)

To get the row percentages do:

prop.table(mytable, 1)

To get the column percentages do:

prop.table(mytable, 2)

Chi-Square:

Create a table with your variables in it or use the one from above.

mytable <- table(varname$Age, varname$Education)

Then run the chi-square test using the command:

chisq.test(mytable)

Linear Regression

Declare and initialize X and Y. Plot the X and Y Axis. Add the regression line, and display the equation.

```
x=varname$Age

y=varname$Education

lm_mod<-lm(y~x)

plot(x,y)

abline(lm_mod)

lm_coef <- round(coef(lm_mod), 3)

 mtext(bquote(y == .(lm_coef[2])*x + .(lm_coef[1])),

+ adj=1, padj=0)
```

Logistic Linear Regression

Y-Dependent Variable 0 and 1.

```
Age.lm = lm(Age ~ Education, data=faithful)

coeffs = coefficients(Age.lm); coeffs
```

Factorial Analysis

Declare and initialize variables you want to perform analysis on.

```
v1=varname$Age

v2=varname$Gender

v3=varname$Beta

v4=varname$Education

m1<- cbind(v1,v2,v3,v4)

cor(m1)

factanal(m1,factors=1) #varimax is default

factanal(m1,factors=1,rotation="promax")
```

prcomp(m1) #signs may depend on platform

##formula interface

factanal(~v1+v2+v3+v4, factors=1,scores="Bartlett")$scores ##Bartlett can be replaced with none, or regression.

Clustering: Hierarchical Agglomerative

d <- dist(varname$Age, method = "euclidean") # distance matrix

fit <- hclust(d, method="ward")

plot(fit)

groups <- cutree(fit, k=5)

rect.hclust(fit, k=5, border="red")

Linear Determination

Rows represent the variable names. The headers are the discrimination.

Iris <- data.frame(rbind(iris3[,,1], iris3[,,2], iris3[,,3]), Sp = rep(c("s","c","v"), rep(50,3)))

train <- sample(1:150, 75) #Creates a sample data set.

table(Iris$Sp[train])

z <- lda(Sp ~ ., Iris, prior = c(1,1,1)/3, subset = train)

predict(z, Iris[-train,])$class

(z1 <- update(z, . ~ . - Petal.W.))

Recursive Partioning -> Decision Tree Analysis

Read the following documentation. The code present on this webpage was tested and it works.
http://www.statmethods.net/advstats/cart.html

Durbin-Watson Test
http://hosho.ees.hokudai.ac.jp/~kubo/Rdoc/library/lmtest/html/dwtest.html

```
## generate two AR(1) error terms with parameter
## rho = 0 (white noise) and rho = 0.9 respectively
err1 <- rnorm(100)

## generate regressor and dependent variable
x <- rep(c(-1,1), 50)
y1 <- 1 + x + err1

## perform Durbin-Watson test
dwtest(y1 ~ x)

err2 <- filter(err1, 0.9, method="recursive")
y2 <- 1 + x + err2
dwtest(y2 ~ x)
```

Additional Resources

http://www.statmethods.net

http://www.r-tutor.com/