Supplementary R Code

Caren Marzban Hoi Yi Ng Corinne Jones

February 11, 2017

1 Introduction to R Basics

R is a programming language and software environment for statistical computing and graphics. There are several graphical user interfaces for R including the basic R console, Rstudio and Tinn-R. Rstudio is recommended for R beginners since it is relatively straightforward to use. In Rstudio, when an incomplete command is entered, a '+' instead of a '<' will be returned. We can use Esc to exit the situation described above.

1.1 Basic Commands

```
# Lines starting with [1] represent outputs.
(57 / 276) ^ 2 + 4.3 * .001
x <- 3  # This sets the variable x to the number 3. `<-' is used to
        # assign the the value 3 to x
x  # Typing the variable, followed by return shows its value.
y <- 1 + 1
log10(y)
log2(y)
sqrt(y)
mean(y)  # Sample mean of y: measures "location" of data.
median(y)  # Sample median of y: another measure of "location."
range(y)  # Gives two numbers, min and max.
range(y)[1]  # First component of range(y), i.e., min.
range(y)[2]  # Second component of range(y), i.e., max.
min(y)  # Another way of getting the minimum value.
max(y)  # Another way of getting the maximum value.
length(y)  # Gives the size of y.
dim(y)  # Gives dimension of y when y is a matrix.
sort(y)  # Sorts all the values in y.
sd(y)  # sample standard deviation of y: measures "spread."
```

Variables can be either number, vector, matrix, dataframe, character, or logical expression.

```
is.vector(x)  # Checking if x is a vector
```

Shown below are a few ways of entering data in R.

```
x <- 1:5
y <- seq(from = 0, to = 10, by = 2)  # seq (0, 10, 2), makes a sequence of
```

```
                                      # numbers from 0 to 10 (including 0 and 10),
                                      # in steps of 2.
y <- c(34, 30, 41, 35, 21)
q()  # Quitting R session if using R on terminal.
```

1.2 Viewing Help Files

```
?median  # Shows function definition and examples.
         # Enter q to exit help page.
??median
help.search("histogram")  # Searches all of R (on your computer).
```

1.3 Acquiring Data

```
# Browsing through available data sets in R
data()  # Space bar will scroll through the data sets.
# q: Enter q to exit if using R on a terminal.
```

Example 1

Reading data from R

```
# Loading data set "cars".
data(cars)  # This loads the data.
?cars  # Gives info on data set; if using R on a terminal,
       # space-bar = scrolls, q = quit.
```

```
cars  # Simply prints all the data onto the screen.
```

```
names(cars)  # Displays the names of the variables.

[1] "speed" "dist"

dim(cars)  # Shows the dimensions of data set.

[1] 50  2
```

```
cars$speed  # Use a dollar sign to select a given column/variable, by name.
cars[, 1]  # Same as above, but selects by column number.
x <- cars$speed  # Selects speed (by name) and
                 # assigns to some variable named x.
x  # Shows speeds.
mean(x)  # Calculates the mean of speed.
sd(x)  # Calculates the standard deviation of speed.
```

Example 2

Reading data from an existing file.

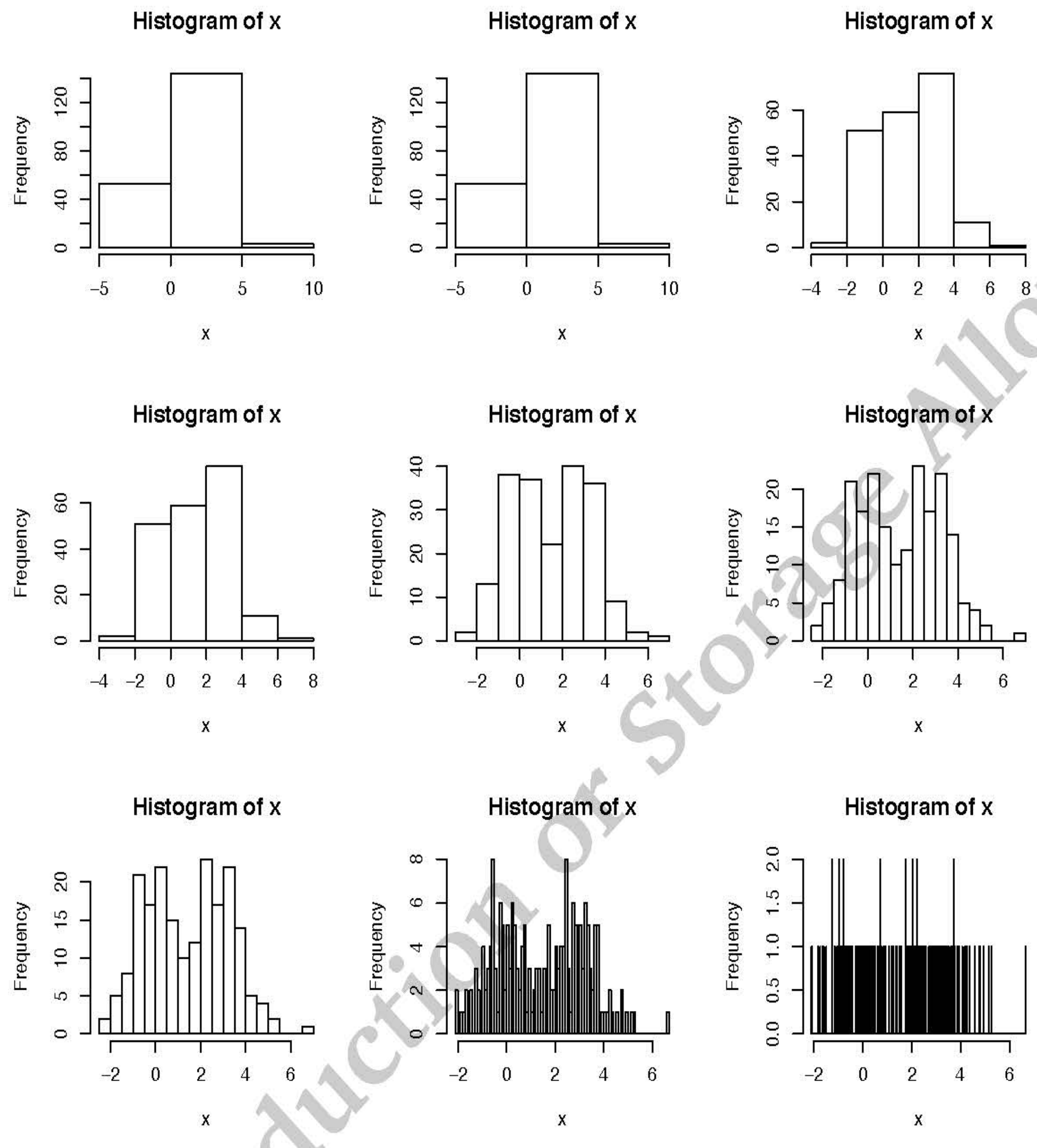
```
# The header=T tells R to ignore the first line/case in the data file.
dat <- read.table(file.choose(), header = T)
```

```
# For reading Excel files, save the file as .csv, and then read it as
dat <- read.csv(file.choose(), header = T)
# In place of file.choose(). a path of the file can be specified.
# Be aware that the path is dependent on the operating system (e.g. linux machines
# use '/' where Window machines use '\\')
# See ?read.csv for details.
```

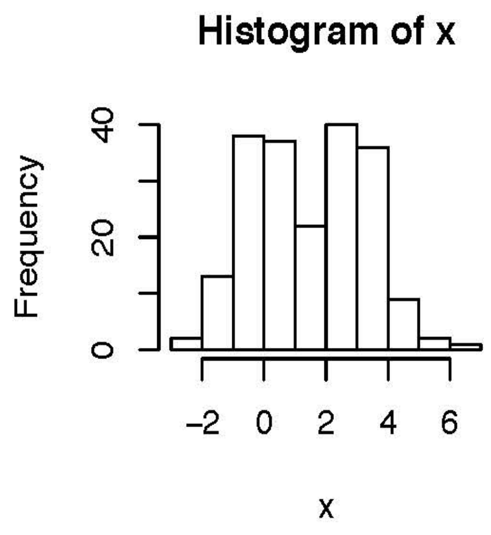
1.4 Plotting Histograms

```
dat <- read.table('hist_dat.txt', header = F)
x <- dat[, 1]  # Selects the first column/variable in the data set named dat to plot.
# This command creates a 3 by 3 grid to display the plots
par(mfrow = c(3,3))

# Histograms with different bin sizes
hist(x, breaks = 2)  # Uninformative
hist(x, breaks = 3)
hist(x, breaks = 4)  # Unimodal and bell-shaped.
hist(x, breaks = 5)
hist(x, breaks = 10)  # Bimodal.
hist(x, breaks = 20)
hist(x, breaks = 30)
hist(x, breaks = 100)  # Bimodal + outlier.
hist(x, breaks = 10000)  # Uninformative.
```



```r
# R decides where to put the breaks. It even decides how many breaks,
# in spite of what is specified by "breaks". To over-ride R's preferences
# and to place the breaks in specific places,
# e.g. -3 -2 -1  0  1  2  3  4  5  6  7  :
hist(x, breaks = seq(-3, 7, by = 1))
```

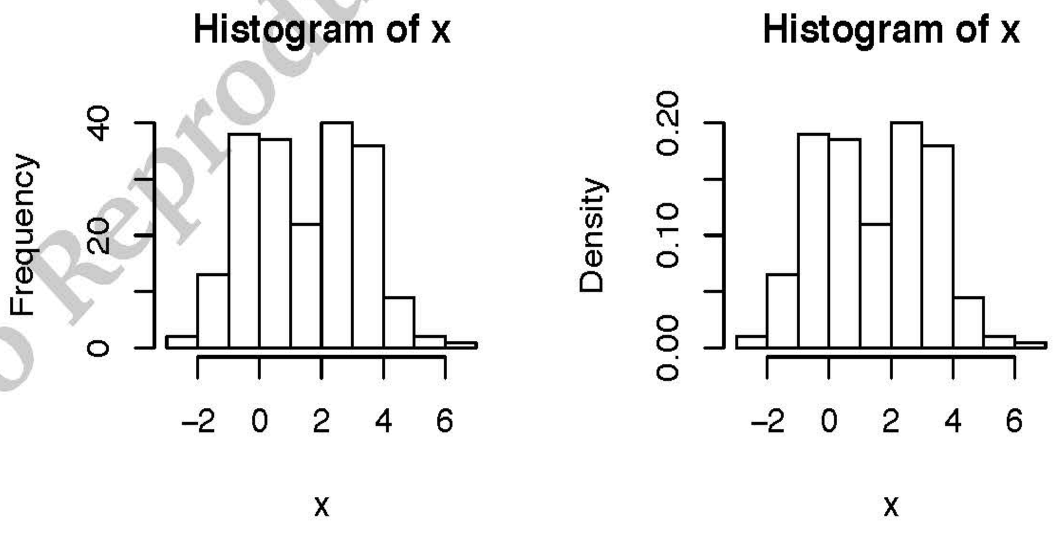


The above suggests that the data/variable x is probably made-up of two different groups. For example, x could be "height," in which case the two "humps" (seen at around breaks = 30 or 100 may be identified as the heights of boys and girls, respectively. This type of analysis is a simple form of datamining, i.e. trying to figure out what is in the data. In general, change the number of breaks, and look for changing patterns that may be of interest.

1.4.1 Making "Density Scale" Histograms

A density scale histogram is a relative frequency histogram where the y-axis represents the frequency in terms of proportion. For example, a frequency of 40 in a total sample of 100 will have density 0.4.

```
par(mfrow = c(1, 2))
hist(x)  # "Regular" histogram.
hist(x, freq = FALSE)  # Density scale histogram.
```



1.5 Outputting Results

```r
# Making a pdf file of a graph and call the file "hello.pdf".
setwd("C:\\Temp")  # Sets the directory to C:\\Temp.
pdf("hello.pdf")  # The file "hello.pdf" is placed in C:\\Temp.
# In Rstudio, this can be done by clicking 'export' on top of the graph and
# choosing a folder to save the graph to.
par(mfrow = c(1, 2))
hist(x)
hist(x, freq = FALSE)
dev.off()  # This makes sure the pdf file is closed.
```