Supplementary R Code

Caren Marzban Hoi Yi Ng Corinne Jones Hassan Nasif

March 24, 2021

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.

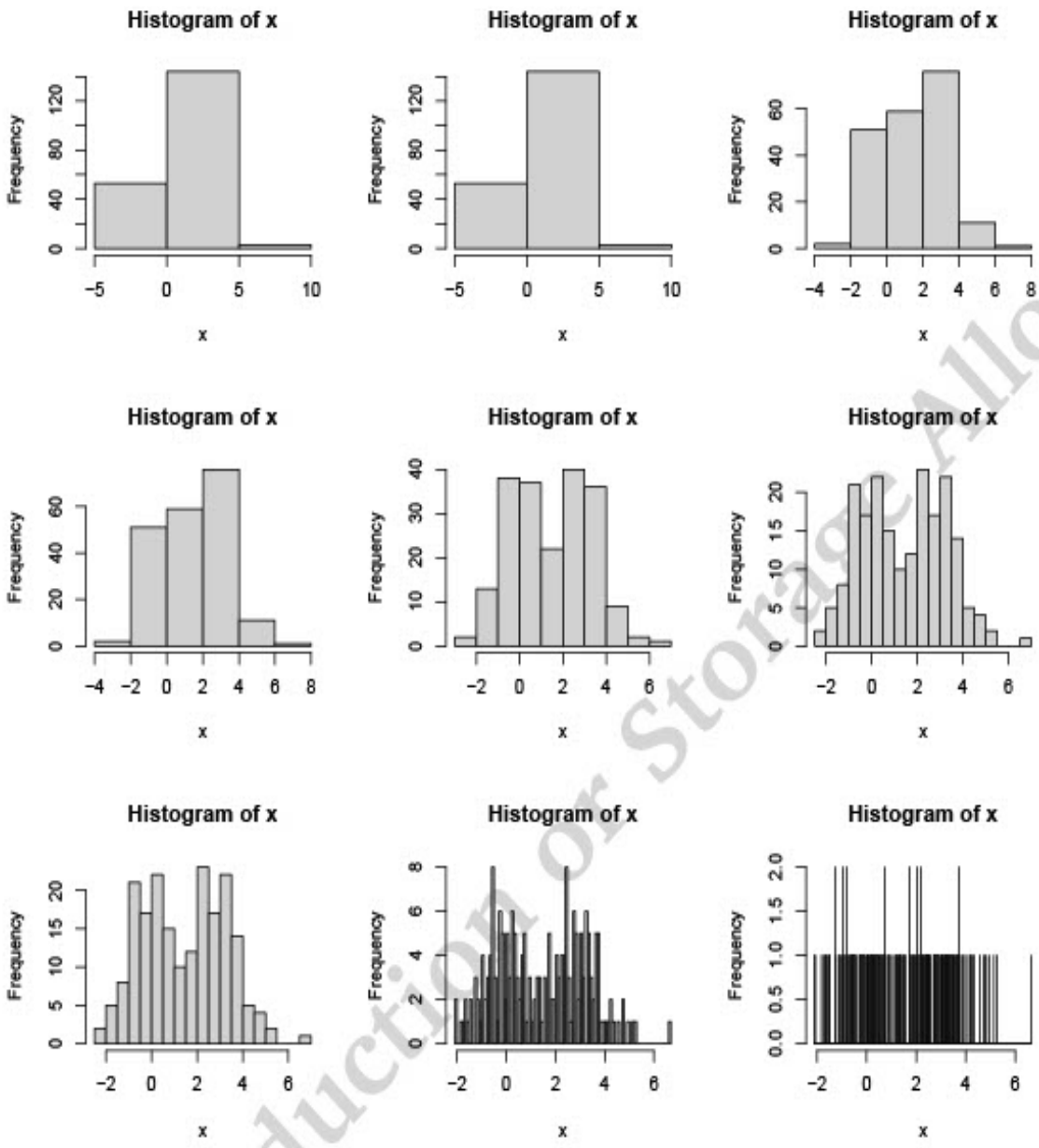
```
# The header=T tells R to ignore the first line/case in the data file.
dat <- read.table(file.choose(), header = T)
```

```
# 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.
```

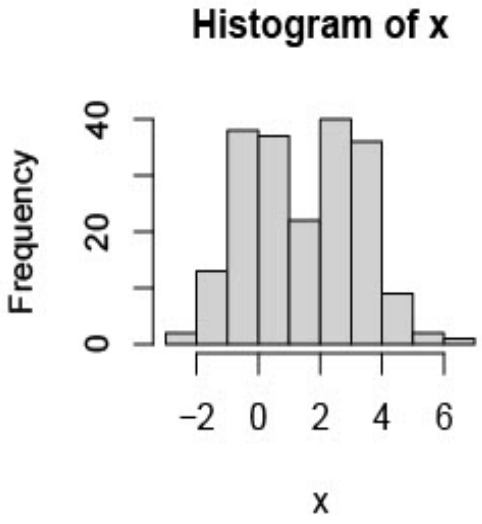
1.4 Plotting Histograms

```
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 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.

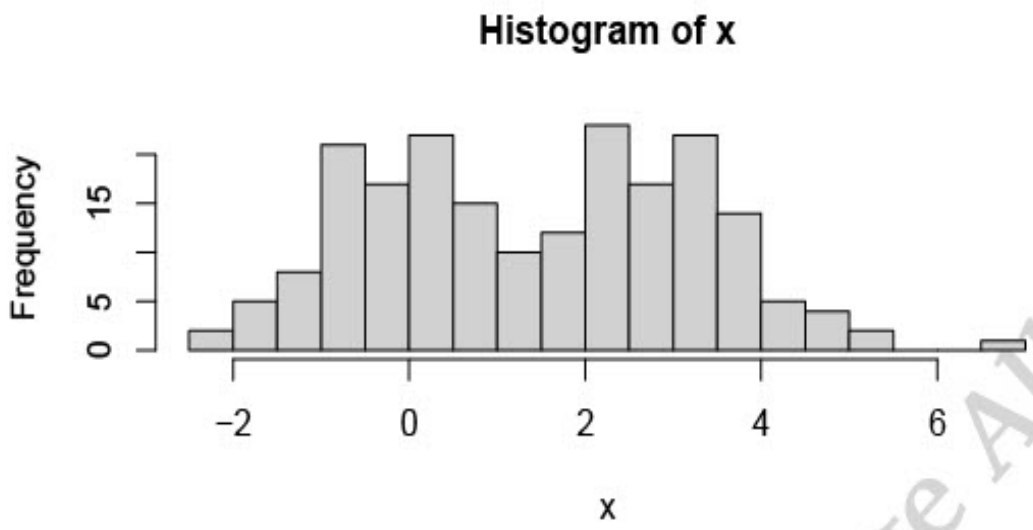
1.4.1 Making "Density Scale" Histograms

One variation on the above (frequency) histogram is the relative frequency histogram, wherein the frequencies on the y-axis are divided by the total number of cases (also called sample size). In this example, all of the bar heights will then be divided by 200. Another variation is obtained by further dividing the relative frequencies by the width of the bars. This version of a histogram is said to be a density (or density-scale) histogram. The advantage of the density histogram is that the area under it is 1. This, then, allows us to interpret the area under the histogram, between any two values of x, as the probability of obtaining an x value in that interval.

To confirm that the area is 1, check this out.

```
H = hist(x, breaks=20)  # Direct the returned values of hist() to H.
names(H)                # Show what is returned (also shown in the help pages for hist().

[1] "breaks"   "counts"   "density"  "mids"     "xname"    "equidist"
```



Now, examine the values of the following:

```
 H$breaks                # Evidently, R decided to let the bin size be 0.5.

 [1] -2.5 -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5
[16]  5.0  5.5  6.0  6.5  7.0

 H$density               # The y-values shown on the density histogram.

 [1] 0.02 0.05 0.08 0.21 0.17 0.22 0.15 0.10 0.12 0.23 0.17 0.22 0.14 0.05 0.04
[16] 0.02 0.00 0.00 0.01
```

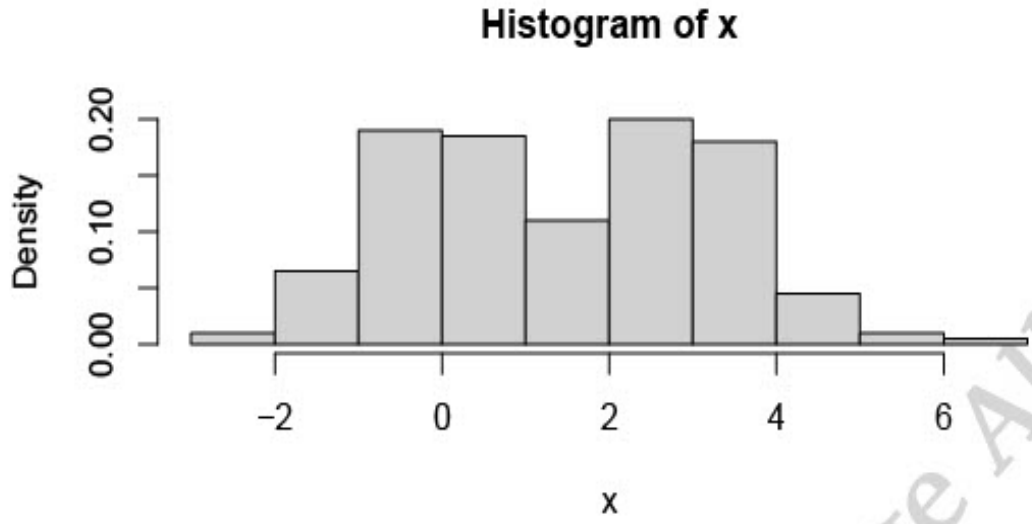
And finally, we can confirm that the total area is 1:

```
 sum(0.5 * H$density)    # = 1

[1] 1
```

The R function hist() does allow for making a density histogram:

```
hist(x, freq = FALSE)  # Density scale histogram.
```



But the reason we did it the "hard way" was to learn more about what R functions return, and how to use them.

1.5 Outputting Results

```
# 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.
```