

# R Practice Answers

*Ryan Safner*

*September 12, 2018*

## Getting Set Up

Before we begin, start a new file with **File** → **New File** → **R Script**. As you work through this sheet in the console in R, also add (copy/paste) your commands that work into this new file. At the end, save it, and run to execute all of your commands at once.

## Creating Objects

1. Create a vector called “me” with two objects, your first name, and your last name. Then call the vector to inspect it. Confirm it is a character class vector.

```
me<-c("Ryan","Safner")
me

## [1] "Ryan"  "Safner"
class(me)

## [1] "character"
```

2. Create a vector called “x” with all the even integers from 2 to 10.

```
x<-c(2,4,6,8,10)
```

3. Find the mean of x with mean()

```
mean(x)

## [1] 6
```

4. Now take the following pdf of random variable  $Y$ :

$y_i$	$p_i$
2	0.50
4	0.25
6	0.25

Calculate the standard deviation “manually” using our table method. You can look at the source code of Lecture 4 for my example.

a. Creating two vectors, one called *y.i* and one called *p.i*, with the data above.

```
y.i<-c(2,4,6)
p.i<-c(0.5,0.25,0.25)
```

b. Merge them into a data frame called *rv* with `data.frame(y.i,p.i)`. Call *rv* to inspect it.

```
rv<-data.frame(y.i,p.i)
rv
```

```
##   y.i  p.i
## 1    2 0.50
## 2    4 0.25
## 3    6 0.25
```

c. Find the expected value of *Y* by taking the sum of each value of *y.i* multiplied by *p.i* with the `sum()` command.

```
sum(rv$y.i*rv$p.i)
```

```
## [1] 3.5
```

d. Creating a new column in *rv* called *deviations*, where you subtract the mean from each *y.i* value. Call *rv* again to make sure it's now there.

```
rv$deviations<-(rv$y.i-3.5)
rv
```

```
##   y.i  p.i deviations
## 1    2 0.50      -1.5
## 2    4 0.25       0.5
## 3    6 0.25       2.5
```

e. Create another column in *rv* called *devsq*, where you square the deviations from part d. Call *rv* again to make sure it's now there.

```
rv$devsq<-(rv$deviations^2)
rv
```

```
##   y.i  p.i deviations devsq
## 1    2 0.50      -1.5  2.25
## 2    4 0.25       0.5  0.25
## 3    6 0.25       2.5  6.25
```

f. Now add another column in `rv` called `weighteddevsq`, where you multiply the squared deviations in part e. by the associated probability `p.i`. Call `rv` again to make sure it's now there.

```
rv$weighteddevsq<-(rv$devsq*rv$p.i)
rv

##   y.i  p.i deviations devsq weighteddevsq
## 1   2 0.50      -1.5  2.25         1.1250
## 2   4 0.25       0.5  0.25         0.0625
## 3   6 0.25       2.5  6.25         1.5625
```

g. Finally, take the sum of `weighteddevsq` to get variance. Square root this to get standard deviation.

```
sum(rv$weighteddevsq)

## [1] 2.75
sqrt(sum(rv$weighteddevsq))

## [1] 1.658312
```

5. The mean height of adults is 65 inches, with a standard deviation of 4 inches. Use the normal distribution to find the probabilities of the following scenarios:

a. Find the probability of someone being *at least* 60 inches tall using `pnorm()`.

```
pnorm(60, mean=65, sd=4, lower.tail=FALSE)

## [1] 0.8943502
```

b. Find the probability of someone being *at most* 60 inches tall.

```
pnorm(60, mean=65, sd=4, lower.tail=TRUE)

## [1] 0.1056498
```

c. Find the probability of someone being between 61 and 69 inches tall. Why is this number familiar?

```
pnorm(69, mean=65, sd=4, lower.tail=TRUE)-pnorm(61, mean=65, sd=4, lower.tail=TRUE)

## [1] 0.6826895
```

d. Find the probability of someone being between 57 and 73 inches tall. Why is this number familiar?

```
pnorm(73, mean=65, sd=4, lower.tail=TRUE)-pnorm(57, mean=65, sd=4, lower.tail=TRUE)

## [1] 0.9544997
```

## Playing with a Data Set

For the following questions, use the `diamonds` dataset, included as part of `ggplot2`.

### 6. Install `ggplot2`

```
install.packages("ggplot2")
```

### 7. Load `ggplot2` with the `library()` command

```
library(ggplot2)
```

### 8. Get the structure of the `diamonds` data.frame. What are the different variables and what kind of data does each contain?

```
str(diamonds)

## Classes 'tbl_df', 'tbl' and 'data.frame':   53940 obs. of  10 variables:
##  $ carat   : num  0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
##  $ cut     : Ord.factor w/ 5 levels "Fair"<"Good"<...: 5 4 2 4 2 3 3 3 1 3 ...
##  $ color   : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<...: 2 2 2 6 7 7 6 5 2 5 ...
##  $ clarity : Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<...: 2 3 5 4 2 6 7 3 4 5 ...
##  $ depth   : num  61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
##  $ table   : num  55 61 65 58 58 57 57 55 61 61 ...
##  $ price   : int  326 326 327 334 335 336 336 337 337 338 ...
##  $ x       : num  3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
##  $ y       : num  3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
##  $ z       : num  2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

### 9. Get summary statistics for `carat`, `depth`, `table`, and `price`

```
summary(diamonds$carat)

##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.2000  0.4000  0.7000  0.7979  1.0400  5.0100
```

```
summary(diamonds$depth)

##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 43.00  61.00  61.80  61.75  62.50  79.00
```

```
summary(diamonds$table)

##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 43.00  56.00  57.00  57.46  59.00  95.00
```

```
summary(diamonds$price)

##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 326      950      2401    3933    5324   18823
```

10. color, cut, and clarity are categorical variables (factors). Use the `table()` command to generate frequency tables for each.

```
table(diamonds$cut)
```

```
##
##      Fair      Good Very Good   Premium    Ideal
##      1610      4906      12082      13791      21551
```

```
table(diamonds$color)
```

```
##
##      D      E      F      G      H      I      J
##  6775  9797  9542 11292  8304  5422  2808
```

```
table(diamonds$clarity)
```

```
##
##      I1      SI2      SI1      VS2      VS1      VVS2      VVS1      IF
##      741    9194 13065 12258  8171  5066  3655  1790
```

Note, you can also use `summary()` to get the counts of each category.

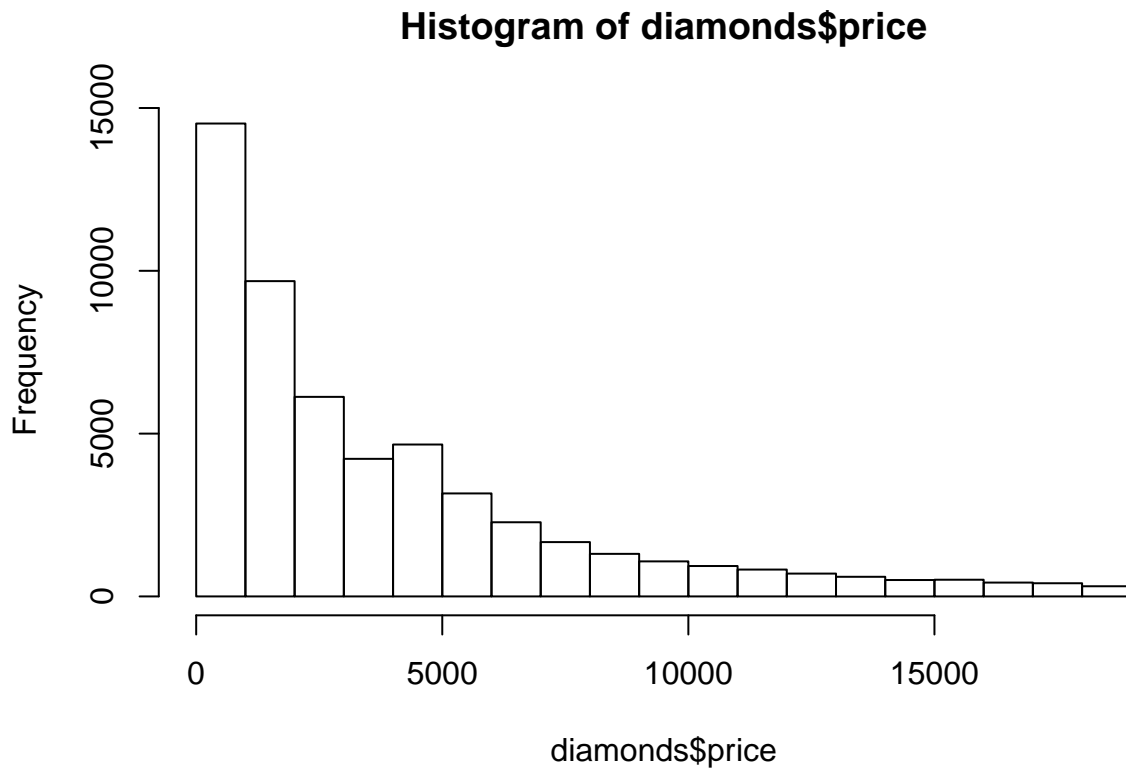
11. Now rerun the `summary()` command on the entire data frame

```
summary(diamonds)
```

```
##      carat      cut      color      clarity
##  Min.   :0.2000   Fair      : 1610   D: 6775   SI1      :13065
##  1st Qu.:0.4000   Good      : 4906   E: 9797   VS2      :12258
##  Median :0.7000   Very Good:12082   F: 9542   SI2      : 9194
##  Mean   :0.7979   Premium   :13791   G:11292   VS1      : 8171
##  3rd Qu.:1.0400   Ideal     :21551   H: 8304   VVS2     : 5066
##  Max.   :5.0100                      I: 5422   VVS1     : 3655
##                                           J: 2808   (Other): 2531
##
##      depth      table      price      x
##  Min.   :43.00   Min.   :43.00   Min.   : 326   Min.   : 0.000
##  1st Qu.:61.00   1st Qu.:56.00   1st Qu.: 950   1st Qu.: 4.710
##  Median :61.80   Median :57.00   Median : 2401   Median : 5.700
##  Mean   :61.75   Mean   :57.46   Mean   : 3933   Mean   : 5.731
##  3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540
##  Max.   :79.00   Max.   :95.00   Max.   :18823   Max.   :10.740
##
##      y      z
##  Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 4.720   1st Qu.: 2.910
##  Median : 5.710   Median : 3.530
##  Mean   : 5.735   Mean   : 3.539
##  3rd Qu.: 6.540   3rd Qu.: 4.040
##  Max.   :58.900   Max.   :31.800
##
```

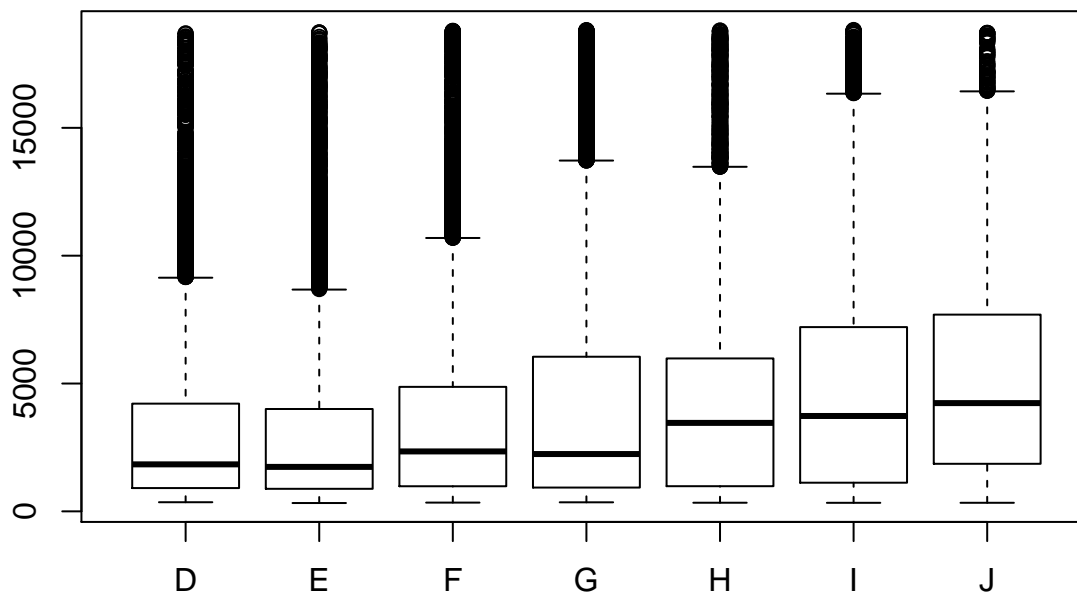
12. Plot a histogram of price.

```
hist(diamonds$price)
```



13. Plot a boxplot of price by diamond color.

```
boxplot(price~color,data=diamonds)
```



## Execute your R Script

Save the R Script you created at the beginning and (hopefully) have been pasting all of your valid commands to. This creates a `.R` file wherever you choose to save it to. Now looking at the file in the upper left pane of *R Studio* look for the button in the upper right corner that says **Run**. Sit back and watch R redo everything you've carefully worked on, all at once.

*# Your R file should look something like this:*

```
# Question 1
me<-c("Ryan","Safner")
me
class(me)

# Question 2
x<-c(2,4,6,8,10)

# Question 3
mean(x)

# Question 4
## part a
y.i<-c(2,4,6)
p.i<-c(0.5,0.25,0.25)

## part b
rv<-data.frame(y.i,p.i)
rv

## part c
sum(rv$y.i*rv$p.i)

## part d
rv$deviations<-(rv$y.i-3.5)
rv

## part e
rv$weighteddevsq<-(rv$devsq*rv$p.i)
rv

## part f
rv$weighteddevsq<-(rv$devsq*rv$p.i)
rv

## part g
sum(rv$weighteddevsq)
sqrt(sum(rv$weighteddevsq))

# Question 5
## part a
pnorm(60, mean=65, sd=4, lower.tail=FALSE)

## part b
pnorm(60, mean=65, sd=4, lower.tail=TRUE)
```

```

## part c
pnorm(69, mean=65, sd=4, lower.tail=TRUE)-pnorm(61, mean=65, sd=4, lower.tail=TRUE)

## part d
pnorm(73, mean=65, sd=4, lower.tail=TRUE)-pnorm(57, mean=65, sd=4, lower.tail=TRUE)

# Question 6
install.packages("ggplot2")

# Question 7
library(ggplot2)

# Question 8
str(diamonds)

# Question 9
summary(diamonds$carat)
summary(diamonds$depth)
summary(diamonds$table)
summary(diamonds$price)

# Question 10
table(diamonds$cut)
table(diamonds$color)
table(diamonds$clarity)

# Question 11
summary(diamonds)

# Question 12
hist(diamonds$price)

# Question 13
boxplot(price~color,data=diamonds)

```