Daniel Markham **710_Fall_2016_Brisbin_and_Riehl**

Assignment Week_3_Sample_Problem due 09/23/2016 at 11:59pm CDT

---

1. (1 point)

This WeBWorK lesson will walk you through the process of conducting an exploratory analysis of a data set using descriptive statistics and graphs.

Download the data set “AmesHousing.csv” **from D2L**. Suppose you have saved the data in the directory C:\Documents\Data Science. Which of these segments of code is most appropriate to load the data into R and attach it?

___

A.
```
ames = read.csv(``C:\Documents\Data Science\AmesHousing.csv'')
attach(ames)
```

B.
```
ames = read.csv(``C:/Documents/Data Science/AmesHousing.csv'')
attach(ames)
```

C.
```
ames = read.csv(C:/Documents/Data Science/AmesHousing.csv)
attach(ames)
```

D.
```
read.csv(``C:/Documents/Data Science/AmesHousing.csv'')
attach(AmesHousing)
```

(Dataset: “Ames, IA Real Estate Data,” submitted by Dean De Cock, Truman State University. Dataset obtained from the Journal of Statistics Education. Accessed 1 July 2015. Used by permission of author.)

Answer(s) submitted:

- `B`

(correct)

---

2. (1 point)

Match each question with the most appropriate type of graph for investigating the question’s answer. Each type of graph will be used exactly once.

[?] 1. Are homes with central air conditioning more likely to have a fireplace than homes without air conditioning?

[?] 2. Do house prices have a bell-shaped distribution?

[?] 3. Do houses with central air conditioning typically cost more than those without air conditioning?

[?] 4. What is the relationship between house prices and the size of the house (measured in square feet)?

A. barplot
B. boxplot
C. scatterplot
D. histogram

Answer(s) submitted:

- `A`
- `D`
- `B`
- `C`

(correct)

---

3. (1 point)

Sale price in dollars is a quantitative variable with so many possible values that we can treat it as continuous. Therefore, a histogram is a good way to explore the shape of its distribution. Type

```
hist(SalePrice)
```

to make a histogram of sales prices.

(Note: 2e+05 means $2 * 10^5$.)

Check all of the adjectives that describe the shape of the distribution of house sale prices.

- A. Left-skewed
- B. Symmetric
- C. Bi-modal
- D. Right-skewed

Answer(s) submitted:

- `D`

(correct)

---

4. (1 point)

It’s often helpful to be able to modify graphs to make them easier to read or to better communicate your ideas. The help pages at ?hist and ?par (short for parameters) show a wide variety of arguments you can include inside the parentheses of *hist()* (or *plot*, *boxplot*, or *barplot*) to customize your graphs. Here are the ones I rely on most often:

Customizing axis and graph labels

- *xlab* changes the label on the x-axis.
- *ylab* changes the label on the y-axis.
- *main* changes the label at the top of the graph.
- Example:
```
hist(SalePrice, xlab = ''Price in Dollars'', ylab = ''Number of Houses'', main = ''My Awesome Histogram'')
```

Customizing axis numbering

- *las=1* causes all the axis numbers to be horizontal. This is usually a good idea, so your readers don't have to turn their heads sideways to read the numbers on the y-axis.
- *cex.axis* changes the font size used for the axis numbers. The default (which is used if you don't include cex.axis as an argument) is 1. Increasing this is usually a good idea if you plan to embed your graph as a figure in a Word document, so your axis numbers are still readable when you shrink the size of the graph.
- Example:
```
hist(SalePrice, las = 1, cex.axis = 1.5)
```

Customizing x and y intervals

- *xlim* sets the lower and upper limits of the x-axis.
- *ylim* sets the lower and upper limits of the y-axis.

- Be careful not to mislead your readers by cutting off data when using xlim and ylim.
- These are most useful when you want to compare two different graphs, using consistent scales for the axes. (We'll see an example of this in Problem 19 in this WeBWorK activity.)
- Example:
```
hist(SalePrice, xlim = c(0,1000000), ylim = c(0, 1200) )
```

[/OK, got it!]
Answer(s) submitted:

- `OK, got it!`

(correct)

---

5. (1 point)

Use
```
mean(SalePrice)
```
to find the mean sale price of houses in the data.

___

Use
```
median(SalePrice)
```
to find the mean sale price of houses in the data.

___

Notice that the mean is higher than the median. This is common in right-skewed data, because the mean is more sensitive to extreme values than the median. The mean tends to be "pulled" toward extreme values.
Answer(s) submitted:

- `180796.10`
- `160000`

(correct)

---

6. (1 point)

Sometimes we want to add lines to graphs to emphasize particular values. For example, let's add vertical lines to our histogram to point out the fact that the mean is higher than the median. Type
```
abline(v = 180796.1, col = ''red'', lwd = 2)
```
- The argument *v*= tells R that the line should be vertical.
  - We could get a horizontal line instead using *h*=.
- *col* tells what color the line should be.
  - This argument is optional. If you leave it out, the default color is black.
  - See **http://www.stat.columbia.edu/ tzheng/files/Rcolor.pdf** for a list of colors you can choose.
- *lwd* tells the width of the line.
  - This argument is optional. If you leave it out, the default width is 1.

Try making a blue line to mark the location of the median, without typing the number 160000.

[/Okay, got it/Show me how]

Notice that the mean is higher than the median. This is common in right-skewed data, because the mean is more sensitive to extreme values than the median. The mean tends to be "pulled" toward extreme values.
Answer(s) submitted:

- `Okay, got it`

(correct)

---

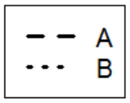
7. (1 point)

When using different colors or symbols on a graph, it's a good idea to include a legend to communicate to readers what the colors represent. Let's add a legend to our histogram to show which color line is the mean and which is the median.

```
legend(''topright'', legend = c(''Mean price'', ''Median pric
```

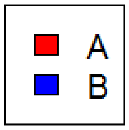
- The first argument tells R where to put the legend. Depending on the graph, it might make more sense to use "top" (centered at the top), "bottomleft", etc.
- *legend* tells what text to put in the legend. Be sure that the order matches the order of your colors in the *col* argument.

Use ?legend to view the help file for the legend function. Use the help file, along with experimentation in R, to match the figures below with the code to create them.

a. 

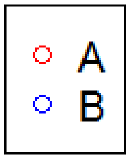
i. legend(“topright”, legend = c(“A”,”B”), lwd=3, lty=c(2,3))
ii. legend(“topright”, legend = c(“A”,”B”), col = c(“red”, “blue”), pch = 21)
iii. legend(“topright”, legend = c(“A”,”B”), fill = c(“red”, “blue”))

[/i/ii/iii]

b. 

i. legend(“topright”, legend = c(“A”,”B”), lwd=3, lty=c(2,3))
ii. legend(“topright”, legend = c(“A”,”B”), col = c(“red”, “blue”), pch = 21)
iii. legend(“topright”, legend = c(“A”,”B”), fill = c(“red”, “blue”))

[/i/ii/iii]

c. 

i. legend(“topright”, legend = c(“A”,”B”), lwd=3, lty=c(2,3))
ii. legend(“topright”, legend = c(“A”,”B”), col = c(“red”, “blue”), pch = 21)
iii. legend(“topright”, legend = c(“A”,”B”), fill = c(“red”, “blue”))

[/i/ii/iii]

Answer(s) submitted:

- `i`
- `iii`
- `ii`

(correct)

---

8. (1 point)

After you get your graph looking the way you want it, you’ll want to save it. There are 3 ways to do this.

1. Right-click on the graph and choose “Save as postscript...”.
   a. You can also choose “Copy as bitmap” and then paste the figure into another document, such as a Word document.
2. Select the graph window in R by left-clicking on it, then click File -> Save As at the top of the R window.
3. You can make a script that automatically saves your graph using the functions pdf() and dev.off().
   a. pdf(“filename.pdf”) starts the creation of a PDF file with the given filename.
   b. Then type the code to create your graph. This can have more than one line: For example, you might use *hist()* to create a histogram and *legend()* to add a legend to it.
   c. dev.off() shuts down the plotting device (in this case, the PDF creator). This is necessary for you to be able to open and view the created file.

Option 3 is especially useful if you want to make a lot of similar plots. For example, columns 68-73 of the AmesHousing data all contain quantitative variables. If we wanted a separate PDF with a histogram of each of these variables, we could use the following code:

```
for(i in 68:73){
    pdf( paste(``C:/Documents/Data Science/Histogram_'', i, ``
    hist( ames[,i], main = paste(``Histogram of'', colnames(ar
    dev.off()
}
```

[/`Okay, got it!`]

Answer(s) submitted:

- `Okay, got it!`

(correct)

---

9. (1 point)

We’ve seen that SalePrice has a right-skewed distribution. Which of these is the best way to reduce right skew in data?

- A. Use a log transformation.
- B. Take the median instead of the mean.
- C. Delete the outliers.
- D. Gather more data.

Answer(s) submitted:

- `A`

(correct)

---

10. (1 point)

Create a new variable, logPrice, containing the natural log (ln) of SalePrice. Then create a new histogram of the transformed sales price.

[/Okay, got it/Show me how]

Notice that the distribution of logPrice is much closer to being symmetric than the distribution of SalePrice was. (logPrice might even be slightly left-skewed.) Let’s explore how well a normal (Gaussian) distribution fits the data.

First, re-scale the y-axis of your histogram using

```
hist(logPrice, prob=T)
```

The argument *prob=T* changes the y-axis so that the total area of the bars of the histogram is 1 (rather than the number of houses in the data). This makes it easier to compare it to a probability distribution (because the total probability of a randomly selected house costing *some* price should add up to 1).

Next, find the mean and standard deviation (sd) of logPrice.
mean = ___
sd = ___

In R, the function *curve* adds a curve to a graph. For our purposes, we need to set two arguments of this function:

- The first argument should be an expression describing the curve we want to plot. It should use *x* as a variable.
- The argument *add=T* will cause the curve to be added to the graph (rather than erasing the current graph and plotting the curve as a new figure).

So, what expression describes the curve we want to plot? The function *dnorm(x, mu, sigma)* gives the height of a normal density curve (with mean *mu* and standard deviation *sigma*) at a value *x*.

- This is the "bell-shaped" curve you've probably seen associated with the normal distribution.
- The area under this curve is used to compute probabilities. For example, the area under the curve between 1 and 3 gives the probability that $1 \le X \le 3$, where X is a normally distributed random variable.

Try adding a normal density curve to your histogram. Use the same mean and standard deviation as in the logPrice data.

For fun, make the curve your favorite color.

[/Okay, got it/Show me how]
The normal density curve isn't a perfect fit for the data, but it's pretty good–certainly better than it would have been for the untransformed data (before we took the log).
Answer(s) submitted:

- `Okay, got it`
- `12.02097`
- `0.4075869`
- `Okay, got it`

(correct)

---

11. (1 point)

House sale prices (in dollars) and the above-ground living area (in square feet) are both quantitative variables, so it makes sense to compute their correlation. Type

```
cor(SalePrice,Gr.Liv.Area)
```

What is the correlation between sale price and living area?
___

This correlation means that the linear relationship between sale price and living area in Ames, Iowa is...

- A. Fairly weak
- B. Extremely strong
- C. Moderately strong

Which of these interpretations of the correlation is correct?

- A. Larger houses tend to cost more.
- B. Increasing the area of a house causes the price to increase.
- C. Larger houses tend to cost less.
- D. We cannot say anything about the direction of the relationship based on the correlation.

Answer(s) submitted:

- `0.7067799`
- `C`
- `A`

(correct)

---

12. (1 point)

A scatterplot is the best type of graph for visualizing the relationship between two continuous variables. There are two basic ways to make a scatterplot in R:

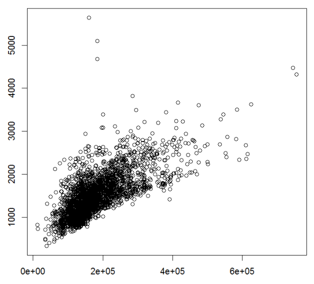
```
plot(x, y)
plot(y ~ x)
```

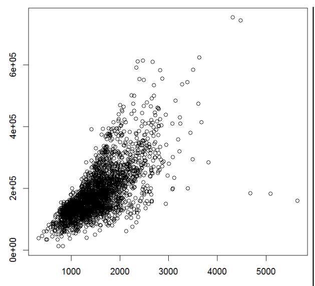
These produce the same graph, but they list the variables in opposite orders.

Regardless of which syntax you use to make the scatterplot, the x variable should be the predictor variable–the variable that is known (or known first), or the variable whose value you can control. The y variable should be the response variable–the variable whose value you're trying to estimate, predict, or model.

If you are told to "plot B as a *function* of A", that means that A should be x and B should be y (because in math, y is often treated as a function of x and written as $y = f(x)$). If you are told to "plot B *versus* A", that usually means that A should be x and B should be y (but sometimes people are sloppy about this).

Plot sales price versus above-ground living area. Select the letter of the matching graph.

A. 

B. 

[/A/B]
Answer(s) submitted:

- `B`

(correct)

---

13. (1 point)

What if we want to explore the relationship between the prices of homes, their living area, *and* whether they have central air conditioning? Central.Air is a categorical variable, so we can assign different colors to each category (in this case, Y or N). Then we can make a scatterplot of the two quantitative variables, but color each point according to whether that home has central air conditioning.

Let's say we want black to represent "Y" (has central air conditioning) and we want red to represent "N" (no central air conditioning). There are 2930 houses in the data set (we can find this using length(Central.Air)), so we can start by making a vector filled with 2930 copies of the word "black":

```
mycolors = rep( ``black'', 2930 )
```

- *rep* repeats the first argument as many times as you tell it in the second argument.

Then we want to change some of the entries in the *mycolors* vector to "red". Which ones? We want to go through each entry of the vector *Central.Air*, check if the entry is "N", and if it is, then change the corresponding entry of *mycolors*.

- This could be done using a **for** loop with an **if** statement nested inside it.
- However, R has a built-in function that's faster: *which()*.
  - The entry inside the parentheses of *which()* should be a vector and a test condition, such as >, <=, or ==.
  - *which()* returns the indices, or locations, of the elements in the vector for which the test condition is TRUE.

Predict the output of the following code. If there are multiple outputs, separate them by commas. Then test your prediction in R.

```
x = c(5, 9, 2, 8, 7)
which( x > 6 )
```

____

Which of the following segments of code will create a vector telling which elements of *mycolors* should be changed to "red"?

[/A/B/C/D]

A. toChange = which(Central.Air == "N")
B. toChange = which(Central.Air = "N")
C. toChange = which(mycolors == "N")
D. toChange = which(mycolors = "red")

Answer(s) submitted:

- `2,4,5`
- `A`

(correct)

---

14. (1 point)

Now we have a vector, *toChange*, containing the indices of all the elements of *mycolors* that should be changed to "red". We could use a **for** loop to change those elements of *mycolors*. However, again there is a faster way.

The first element of *toChange* turns out to be 83. So we want to change the 83rd element of *mycolors* to be "red". We know that we could do that using square bracket notation:

```
mycolors[83] = "red"
```

But square bracket notation also works for vectors of indices. So we can change all of the *toChange* elements of *mycolors* at once, using the code

```
mycolors[ toChange ] = "red"
```

Then make the scatterplot using the new colors:

```
plot( Gr.Liv.Area, SalePrice, col = mycolors )
```

We can see that houses without central air conditioning tend to be less expensive than similar-sized houses with central air conditioning. Also, all of the very large houses have central air conditioning.

Try to add a legend to the scatterplot to explain the meaning of the colors.

[/Okay, got it/Show me how]
Answer(s) submitted:

- `Okay, got it`

(correct)

---

15. (1 point)

In problems 13 and 14 in this lesson, you saw how to use the *which()* function to assign colors to represent different values of a categorical variable. In this exercise, you'll see another use of *which* to compute the proportion of houses meeting certain criteria.

Suppose we want to count the number of houses in the data set that cost between \$200,000 and \$300,000 (not counting the houses that cost exactly \$200,000 or \$300,000). Try writing code to do this using control flow structures. (Note that to test the condition "price is between 200000 and 300000", you need to test “price is greater than 200000” AND “price is less than 300000”. In R, “AND” is represented by &.)

[/Okay, got it/Show me how]
Control flow structures work fine to solve this problem, but there's a faster way. First, use *which* to find the indices of the entries in SalePrice that meet the criteria:

```
    meetCriteria = which( (SalePrice > 200000) & (SalePrice < 300000) )
```

Then use *length* to find the number of indices in the *meetCriteria* vector:

```
    length(meetCriteria)
```

This gives the number of houses that meet the criteria. It should match the number that you got from the control flow structure.

How many houses cost between $200,000 and $300,000 (exclusive–that is, not counting the endpoints)?

___

To find the proportion of houses that meet the criteria, divide this number by the total number of houses, which you can find using *length(SalePrice).*

What proportion of houses cost between $200,000 and $300,000?

___

It is also possible to combine *length* and *which* into a single line of code:

```
    length( which( (SalePrice > 200000) & (SalePrice < 300000) ) )
```

Answer(s) submitted:

- `Okay, got it`
- `619`
- `0.2112628`

(correct)

---

16. (1 point)
Barplots are the most appropriate way to graph categorical variables. They can either be used to plot quantitative information for different categories (such as the mean price of houses with and without air conditioning), or to plot the frequency of each category.

Fireplaces is a discrete, numerical variable that tells how many fireplaces each house has. Let's make a bar plot of the number of houses with 0, 1, 2, 3, or 4 fireplaces. First, use the *table* function to count the number of houses in each category:

```
    counts = table(Fireplaces)
```

How many houses in the data set had 1 fireplace?

___

Then make the barplot:

```
    barplot( counts )
```

Try adding appropriate labels to the x and y axes and to the overall graph. (Hint: It might help to look back at problem 4 in this activity.)

[/Okay, got it/Show me how]
Answer(s) submitted:

- `1274`
- `Okay, got it`

(correct)

---

17. (1 point)
What if we want a graphical representation of 2 categorical variables at the same time? Barplots can be used for that too!

Let's make a bar plot of the number of houses with 0, 1, 2, 3, or 4 fireplaces, and with or without air conditioning. First, use the *table* function to count the number of houses in each category:

```
    counts = table(Central.Air, Fireplaces)
```

How many houses in the data set had 1 fireplace but no central air conditioning?

___

Then make the barplot:

```
    barplot( counts, col = c(''red'', ''blue'') )
```

You can choose whatever colors you want. Their order will correspond to the order of categories listed as the row names of *counts*. So in this example, red corresponds to N, or no air conditioning.

Try adding a legend to the graph to explain the colors. (Hint: It might help to look back at problem 7 in this activity.)

[/Okay, got it/Show me how]
Answer(s) submitted:

- `31`
- `Okay, got it`

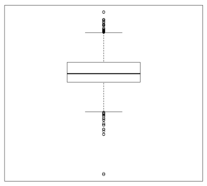
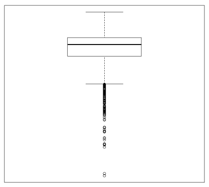
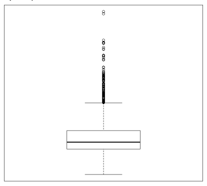
(correct)

---

18. (1 point)
Boxplots are useful for giving an overview of the median and amount of variability in a quantitative variable. Type

```
    boxplot(SalePrice)
```

Which of these shows the boxplot of house sales prices?

(A)  (B)  (C) 

[/A/B/C]

We can see many outliers on the high end of the boxplot. This indicates that the prices are right-skewed, which agrees with what we saw in the histogram. (Right-skewed distributions are common when dealing with money, because there is a fixed lower bound–it's impossible for a house to cost less than \$0–but no upper limit.)

Because boxplots show less detail than histograms, they are especially good for comparing multiple groups. The following code lets us compare sale prices for houses with and without central air conditioning:

```
    boxplot( SalePrice ˜ Central.Air )
```

(On most keyboards, the tilde (˜) key is located next to the number 1.)

Which group of houses has more variability in price?

- A. Houses with central air conditioning, because the center line of the boxplot is higher.
- B. Houses without central air conditioning, because the boxplot is wider and has more outliers.
- C. Houses with central air conditioning, because the boxplot is wider and has more outliers.
- D. Houses without central air conditioning, because the center line of the boxplot is higher.

In this case, all of the sale prices we wanted to compare were in a single vector (SalePrice), and we used another vector (Central.Air) to tell which elements belonged to which group. Sometimes, we will instead want to compare values for which each group is stored in a different vector.

For example, suppose we want to compare the area (in square feet) on the first floor versus the second floor of homes. These values are stored in the variables X1st.Flr.SF and X2nd.Flr.SF. We can use the following code to make the boxplots:

```
    boxplot( X1st.Flr.SF, X2nd.Flr.SF )
```

Notice that for groups stored in *different* variables we use a comma, while for groups stored in the *same* variable we used a tilde.

We can make the x-axis labels for each group more informative using the argument *names*:

```
    boxplot( X1st.Flr.SF, X2nd.Flr.SF, names=c(``1st floor'',''2n
```

Notice that using boxplots lets us compare the medians and amount of variability between the two groups, but it ignores the relationship between the variables–the fact that each value in X1st.Flr.SF represents the first-floor area of one house, which has a corresponding second-floor area value in X2nd.Flr.SF.

What kind of graph would be best for examining the one-to-one relationship between X1st.Flr.SF and X2nd.Flr.SF?

- A. Barplot
- B. Boxplot
- C. Histogram
- D. Scatterplot

Answer(s) submitted:

- `C`
- `C`
- `D`

(correct)

---

19. (1 point)

Sometimes we want to compare quantitative variables for two different groups in more detail than a boxplot allows. In this case, we can use either stacked histograms or side-by-side histograms.

Let's use stacked histograms to compare the sale prices of houses with and without central air conditioning. The first step is to create a new variable containing the sale prices of houses with A/C, and a separate variable containing the sale prices of houses without A/C. Try doing this with the *which* function.

[/Okay, got it/Show me how]

Now we need to prepare the graphing window to plot two graphs in the same window, one above the other. Type

```
    par( mfrow = c( 2, 1 ) )
```

This prepares the graphing window to plot graphs in 2 rows and 1 column.

Now plot the histograms. To enable comparison of the distributions, use *xlim* to set the same span of the horizontal axis for each graph.

[/Okay, got it/Show me how]

When you're done plotting two graphs in one window, you can type

```
    par( mfrow = c( 1, 1 ) )
```

to prepare the graphing window to plot 1 graph at a time (that is, plot 1 row and 1 column of graphs).

Answer(s) submitted:

- `Okay, got it`
- `Okay, got it`

(correct)

---

20. (1 point)

In the previous problem, you made stacked histograms to compare the distributions of sale prices for houses with and without air conditioning. In this problem, we'll use side-by-side histograms for the same comparison.

The function for creating side-by-side histograms is in the *plotrix* package. The first time you use this package, you'll need to install it by typing

```
    install.packages( "plotrix")
```

(If R asks you which server to download it from, you can choose any of the options, just like when you installed R. I usually choose 0-Cloud.)

After the package is installed, you need to load it by typing

```
    library( plotrix )
```

You'll need to do this every time you start a new session of R when you want to make side-by-side histograms.

To make the histogram, use

```
    multhist( list(priceWithAC, priceNoAC), col=c("blue", "
```

Notice that because the number of houses with air conditioning is so much larger than the number of houses without it, it's hard to compare the proportions of houses at each price level. We can correct this by re-scaling the y-axis to refer to probabilities rather than absolute numbers of houses. We do this by setting "frequency" (absolute counts) to FALSE:

```
    multhist( list(priceWithAC, priceNoAC), freq = F, col=c("blue'
```

What does this side-by-side histogram shows us?

- A. There are no houses without A/C that cost more than 180,000 dollars.
- B. Houses without A/C are more likely to cost less than 180,000 dollars, compared to houses with A/C.
- C. There are more houses without A/C that cost less than 180,000 dollars, compared to houses with A/C.

Try creating a legend to explain the colors of this graph.

[/Okay, got it/Show me how]

Answer(s) submitted:

- `B`
- `Okay, got it`

(correct)