Pair Programming Exercise 1, DSE5001

HDS

2024-08-06

Pair Programming Exercise 1, DSE5001

HD Sheets, August 6, 2024 checked 1/3/2025

#Student Info

Your name:Ryan Waterman Your team-mates names:Trinity Tobin, Nicholas Perry

#Libraries

R uses libraries or packages of additional code to add more functions and abilities to R. There are over 19,000 packages available

```
library("ggplot2")
library("tidyr")
```

Working with built-in example data

Most R packages have example “lab rat” data included for you to work with and learn from (Python has this as well).

This data is loaded in various formats, most of them are data frames, which are structured like an SQL “table” or an Excel “workbook”

These are “flat data tables” where each column is a variable, a measurement and each row is an observation (aka an individual, or an event, or a recording).

These flat data tables are the most common form of data storage you will see, other many others are possible. We’ll start here

We can see the available example data sets in R using the command “data()”

This is a function call, which we can tell because it ends with closed parenthesis ().

```
data()
```

#The Texas Housing Set

This is a pretty big data set, but we are going to have a look despite that. It came with the ggplot2 package

```
data(txhousing)
```

#Starting to learn about our data

There are a bunch of R commands that help us learn about what we have in a data structure we aren’t initially familiar with

Look at these and remember them….

The first thing we want to do is see what the structure of the data set is like

use the str() function, and send it the name of the data

```
str(txhousing)
```

```
## tibble [8,602 × 9] (S3: tbl_df/tbl/data.frame)
##  $ city     : chr [1:8602] "Abilene" "Abilene" "Abilene" "Abilene" ...
##  $ year     : int [1:8602] 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 ...
##  $ month    : int [1:8602] 1 2 3 4 5 6 7 8 9 10 ...
##  $ sales    : num [1:8602] 72 98 130 98 141 156 152 131 104 101 ...
##  $ volume   : num [1:8602] 5380000 6505000 9285000 9730000 10590000 ...
##  $ median   : num [1:8602] 71400 58700 58100 68600 67300 66900 73500 75000 64500 59300 ...
##  $ listings : num [1:8602] 701 746 784 785 794 780 742 765 771 764 ...
##  $ inventory: num [1:8602] 6.3 6.6 6.8 6.9 6.8 6.6 6.2 6.4 6.5 6.6 ...
##  $ date     : num [1:8602] 2000 2000 2000 2000 2000 ...
```

This result tells us this is

- A data type called a “tibble” which is variant of the R dataframe. You can just think of it as a data frame, it has some extra
  features, but is otherwise identical
- The “dimension” is [8602 x 9], meaning it has 8602 rows and 9
  columns
- The variables are city, year,month, sales, volume,median, listings, inventory and date

  City is a “chr” which means character string, or text

  Most other variables are “num”, meaning real numbers, one is an “int” meaning an integer

We can see the help listing in R for this data using the help() function

```
help("txhousing")
```

```
## starting httpd help server ... done
```

#Question/Action 1

There is a data set called trees, we’ll load it and then answer some questions about it

```
data(trees)
```

What type of structure is this? - This is a ‘data.frame’ structure.

What are the variables within it? (ie columns) - Girth, height, and volume.

What data types are the columns? - All data types are num

How many rows does it have? - 31 rows

(Add cells as needed to answer these questions)

You can cut and paste the code from earlier examples! Cut and paste is your friend, just like google is! Bad coffee…is not your friend! :)

```
str(trees)
```

```
## 'data.frame':    31 obs. of  3 variables:
##  $ Girth : num  8.3 8.6 8.8 10.5 10.7 10.8 11 11 11.1 11.2 ...
##  $ Height: num  70 65 63 72 81 83 66 75 80 75 ...
##  $ Volume: num  10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 ...
```

```
help(trees)
```

#Head and Tail

head() and tail() will show us just a few rows and all variables, so we can get a quick look at the data

The default is 6 rows, but we can specify more or less

Notice we can see the data types with head() or tail()

```
head(txhousing)
```

```
## # A tibble: 6 × 9
##   city     year month sales   volume median listings inventory  date
##   <chr>   <int> <int> <dbl>    <dbl>  <dbl>    <dbl>     <dbl> <dbl>
## 1 Abilene  2000     1    72  5380000  71400      701       6.3 2000
## 2 Abilene  2000     2    98  6505000  58700      746       6.6 2000.
## 3 Abilene  2000     3   130  9285000  58100      784       6.8 2000.
## 4 Abilene  2000     4    98  9730000  68600      785       6.9 2000.
## 5 Abilene  2000     5   141 10590000  67300      794       6.8 2000.
## 6 Abilene  2000     6   156 13910000  66900      780       6.6 2000.
```

```
tail(txhousing,8)
```

```
## # A tibble: 8 × 9
##   city           year month sales   volume median listings inventory  date
##   <chr>         <int> <int> <dbl>    <dbl>  <dbl>    <dbl>     <dbl> <dbl>
## 1 Wichita Falls  2014    12   109 13883668 103800      821       7   2015.
## 2 Wichita Falls  2015     1    71  7519961  82100      829       7.2 2015
## 3 Wichita Falls  2015     2   100 11646765  94000      795       6.8 2015.
## 4 Wichita Falls  2015     3   152 16716584  89200      818       6.8 2015.
## 5 Wichita Falls  2015     4   129 15482194 105300      760       6.4 2015.
## 6 Wichita Falls  2015     5   174 19188181 100000      776       6.4 2015.
## 7 Wichita Falls  2015     6   143 18820752 118800      770       6.2 2015.
## 8 Wichita Falls  2015     7   172 23850905 116700      811       6.5 2016.
```

We can view the whole data set using View()

```
View(txhousing)
```

#Question/Action 2

Why should you get in the habit of using head() or tail() instead of view()?

- Head and tail load a subset of the data that will be indicative enough of the data structure and variables to understand the dataset. Loading the whole dataset with view is computationally expensive and time consuming.

Class() and dim() will tell us the data type and size quickly

```
class(txhousing)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

The last entry indicates this is a variation on a dataframe

```
dim(txhousing)
```

```
## [1] 8602    9
```

8602 rows and 9 variables

#Summary

Most data types have a summary function that will tell us something about data

```
summary(txhousing)
```

```
##      city                year          month             sales        
##  Length:8602        Min.   :2000   Min.   : 1.000   Min.   :   6.0  
##  Class :character   1st Qu.:2003   1st Qu.: 3.000   1st Qu.:  86.0  
##  Mode  :character   Median :2007   Median : 6.000   Median : 169.0  
##                     Mean   :2007   Mean   : 6.406   Mean   : 549.6  
##                     3rd Qu.:2011   3rd Qu.: 9.000   3rd Qu.: 467.0  
##                     Max.   :2015   Max.   :12.000   Max.   :8945.0  
##                                                     NA's   :568     
##      volume              median          listings       inventory      
##  Min.   :8.350e+05   Min.   : 50000   Min.   :    0   Min.   : 0.000  
##  1st Qu.:1.084e+07   1st Qu.:100000   1st Qu.:  682   1st Qu.: 4.900  
##  Median :2.299e+07   Median :123800   Median : 1283   Median : 6.200  
##  Mean   :1.069e+08   Mean   :128131   Mean   : 3217   Mean   : 7.175  
##  3rd Qu.:7.512e+07   3rd Qu.:150000   3rd Qu.: 2954   3rd Qu.: 8.150  
##  Max.   :2.568e+09   Max.   :304200   Max.   :43107   Max.   :55.900  
##  NA's   :568         NA's   :616      NA's   :1424    NA's   :1467    
##       date        
##  Min.   :2000  
##  1st Qu.:2004  
##  Median :2008  
##  Mean   :2008  
##  3rd Qu.:2012  
##  Max.   :2016  
##
```

We can get a listing of the columns in a dataframe

```
colnames(txhousing)
```

```
## [1] "city"      "year"      "month"     "sales"     "volume"    "median"   
## [7] "listings"  "inventory" "date"
```

#Question/Action 3

show the first ten entries in trees

```
head(trees,10)
```

```
##    Girth Height Volume
## 1    8.3     70   10.3
## 2    8.6     65   10.3
## 3    8.8     63   10.2
## 4   10.5     72   16.4
## 5   10.7     81   18.8
## 6   10.8     83   19.7
## 7   11.0     66   15.6
## 8   11.0     75   18.2
## 9   11.1     80   22.6
## 10  11.2     75   19.9
```

Generate the summary

```
summary(trees)
```

```
##      Girth           Height       Volume
##  Min.   : 8.30   Min.   :63   Min.   :10.20
##  1st Qu.:11.05   1st Qu.:72   1st Qu.:19.40
##  Median :12.90   Median :76   Median :24.20
##  Mean   :13.25   Mean   :76   Mean   :30.17
##  3rd Qu.:15.25   3rd Qu.:80   3rd Qu.:37.30
##  Max.   :20.60   Max.   :87   Max.   :77.00
```

What is mean girth (diameter) of trees in this set? 13.25

What is the height of the smallest and shortest trees?

```
#Height of the shortest tree can be found by the min height in the summary, or using the functio
n below:
min(trees["Height"])
```

```
## [1] 63
```

```
#The height of the smallest tree means the height of the tree with the lowest volume. This can b
e found with the function, below:
volume_vector = as.vector(trees$Volume)
smallest_tree <- trees$Height[match(min(trees$Volume), volume_vector)]
smallest_tree
```

```
## [1] 63
```

Look at the View() for trees. Is it any more useful than head()?

```
View(trees)
```

- In this specific the view function shows nearly the same amount of data as the head function, therefore it does not provide much additional value.

We can access a single variable (column) of a dataframe by using the name of the dataframe followed by a dollar sign and then the name of the column

So txhousing$sales is the list of all the monthly sales from the table

We can feed this into functions in R to find things out, like the mean and standard deviation

```
mean(txhousing$sales,na.rm=TRUE)
```

```
## [1] 549.5646
```

The line na.rm=TRUE means to remove NA values, R uses NA to indicate missing data. Some cities probably had no reported data in some months

```
#computing a standard deviation- another measure of spred

sd(txhousing$sales,na.rm=TRUE)
```

```
## [1] 1110.737
```

#Question/Action 4

Find the standard deviation of the tree height variable

```
sd(trees$Height, na.rm=TRUE)
```

```
## [1] 6.371813
```

You may have to look up how to calculate a standard deviation in R

Graphics

#Histograms

Histograms show us the distribution of values in a data set

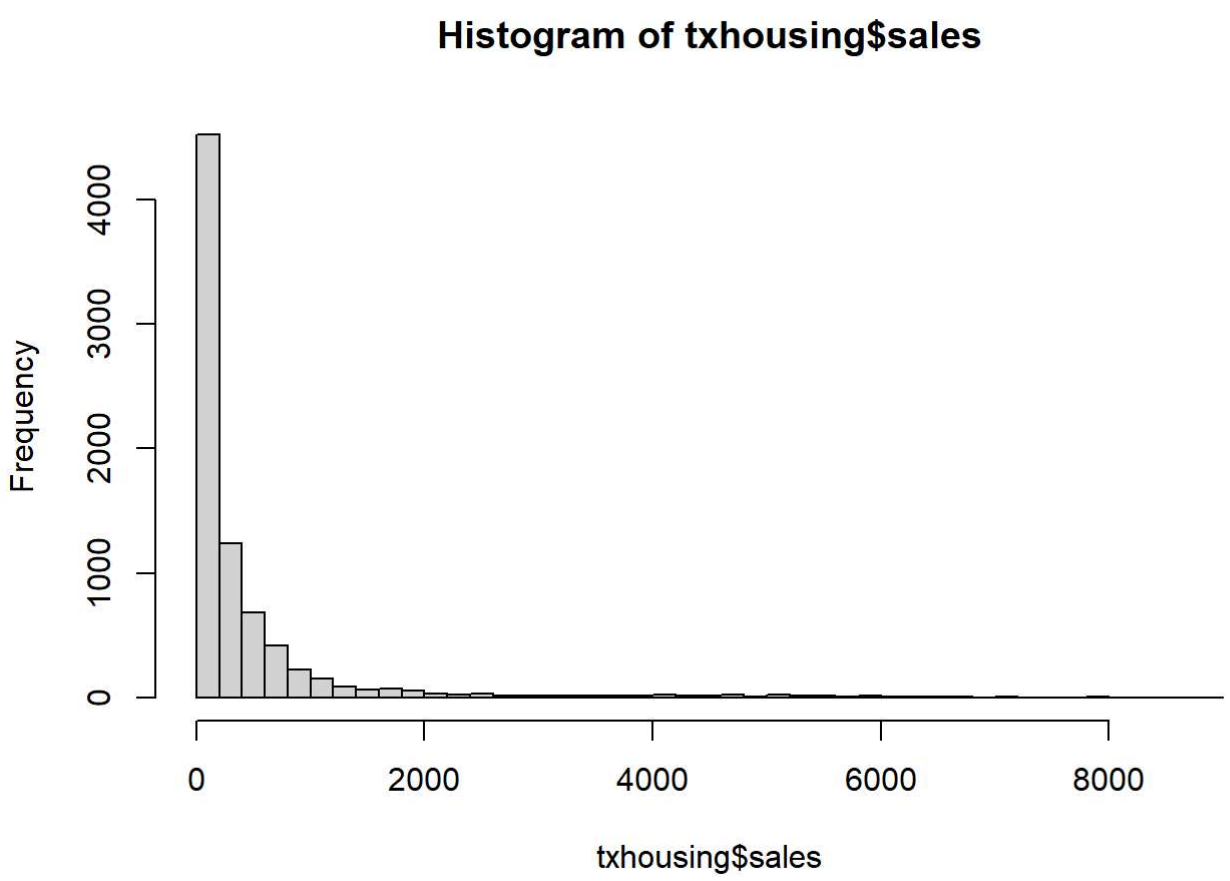
Here is a basic histogram using the built-in hist() function

This plot shows the number of instances (rows) at each value of “sales”

This is a “frequency distribution” or just a distribution

We will talk more later about distributions

```
hist(txhousing$sales,breaks=50)
```



A fancier histogram using ggplot

the first item in ggplot is the name of the data frame

in aes() we specify the variables to plot, and maybe to use for color coding

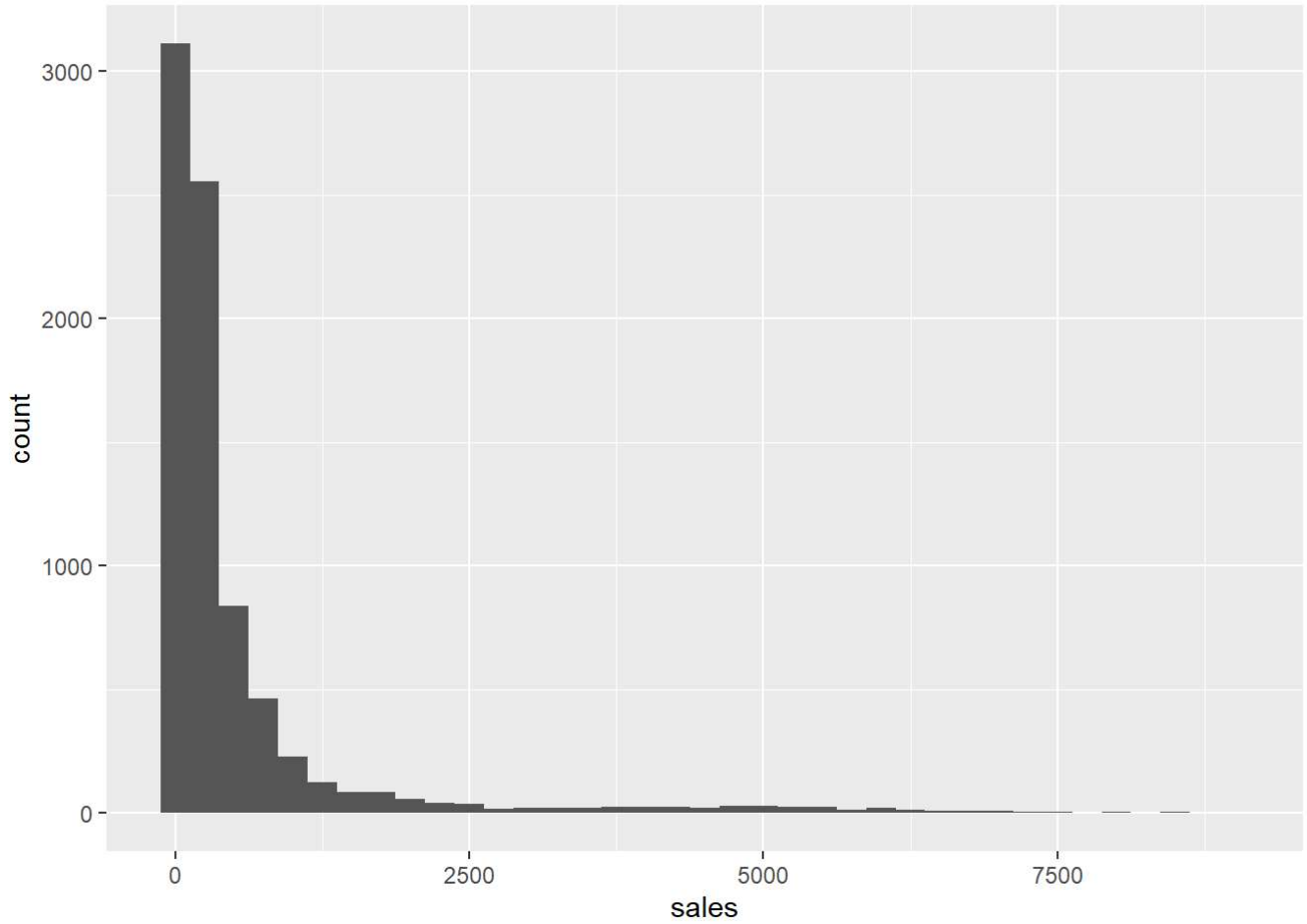
the +geom_histogram means to use the data and variables specified in a histogram plot

In ggplot, we always specify the dataframe and the aesthetic (aes) and then add other pieces to create a complex image

ggplot has many, many options

```
ggplot(txhousing,aes(x=sales))+geom_histogram(binwidth=250)
```

```
## Warning: Removed 568 rows containing non-finite outside the scale range
## (`stat_bin()`).
```



The sales distribution does not look “normal” or “bell curve”, or “gaussian” - (these are all synonyms)

Why would that be?

```
View(txhousing)
```

- Based on the y-axis of the chart, the count of a given number of sales is the measured variable. In this case, it is reasonable to assume that smallers numbers of sales have a greater number of occurrences. Also, considering the date range of the data, there were financial factors that significantly affected the housing market, which minimize the count of large sale months (i.e. the data is skewed low).

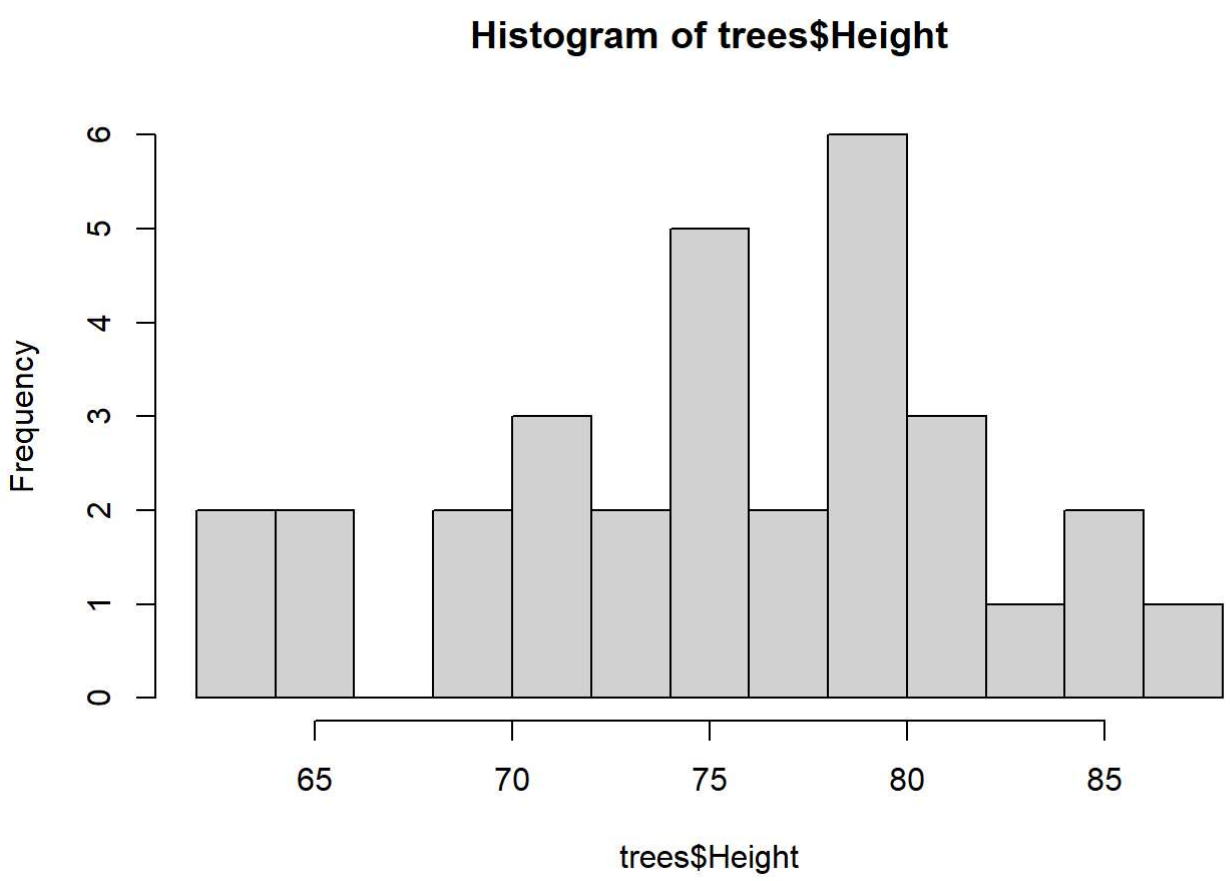
#Question/Action 5

Produce both types of histograms of tree heights

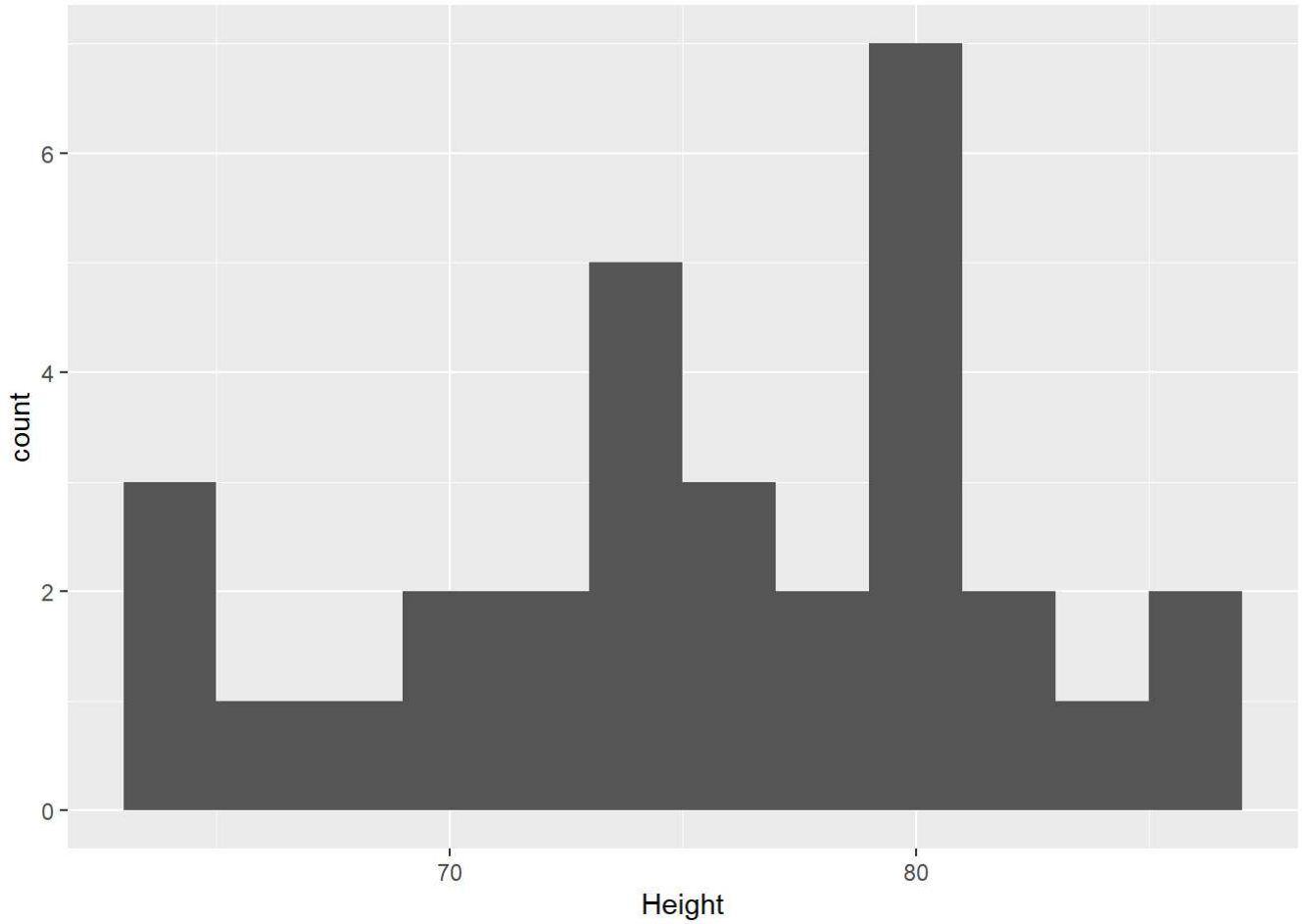
Does this look like a bell curve?

- This data set loosely looks like a bell curve, but to be certain, a larger data set would be required.

```
hist(trees$Height,breaks=10)
```




```
ggplot(trees,aes(x=Height))+geom_histogram(binwidth=2)
```



#Box Plots

These plots show

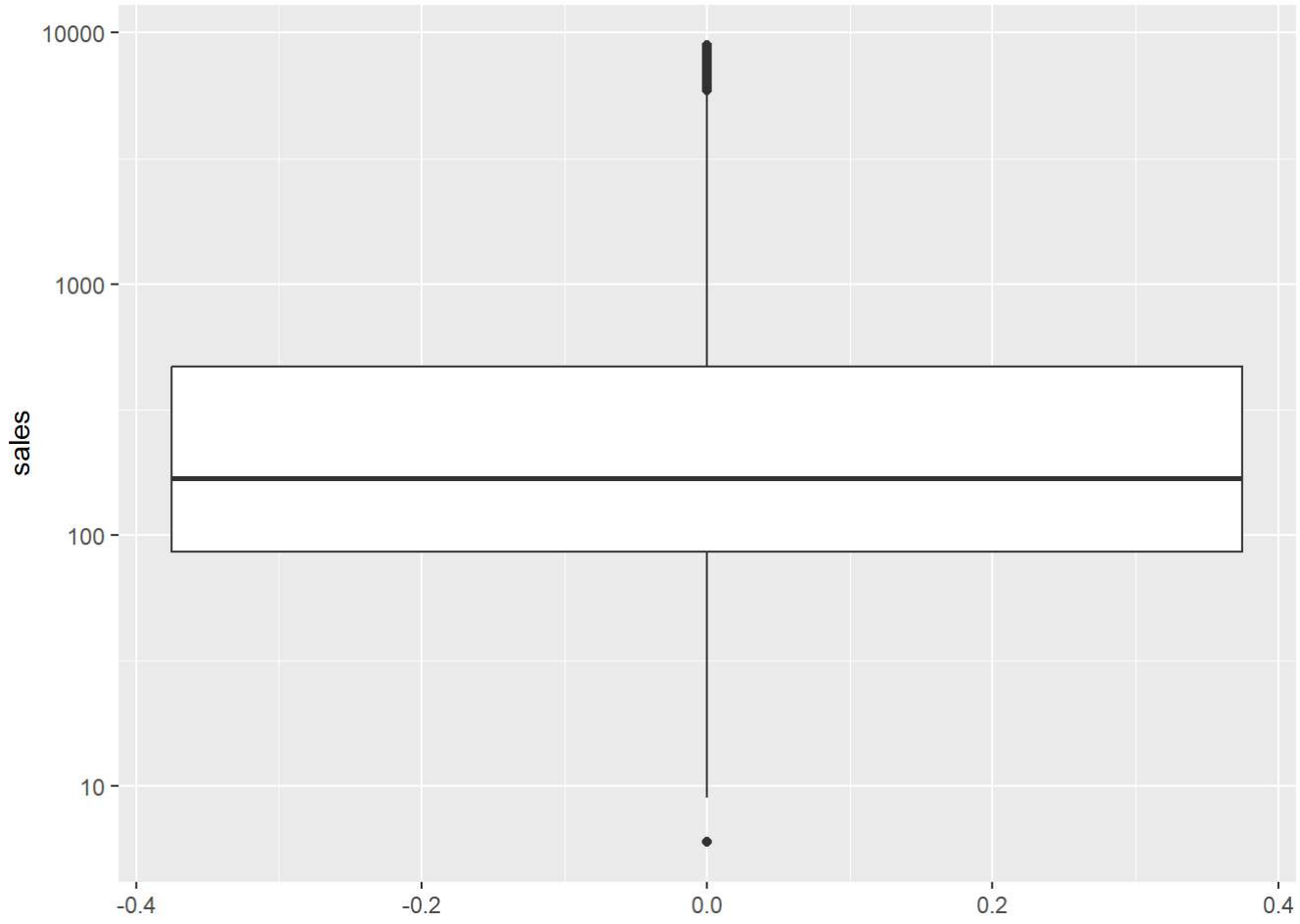
-the median value (center bar) -the 25% and 75% quantiles, the ends of the box -the 5% and 95% upper and lower bounds (the “whiskers”) -Outliers, which are dots

In the boxplot below, I changed the y-scale so it is log-scaled (or “octaved”), which makes it easier to see small values

The need for a log scale here indicates the data probably is not “normal” or we don’t have a bell curve distribution

```
ggplot(txhousing,aes(y=sales))+geom_boxplot()+scale_y_continuous(trans='log10')
```

```
## Warning: Removed 568 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```

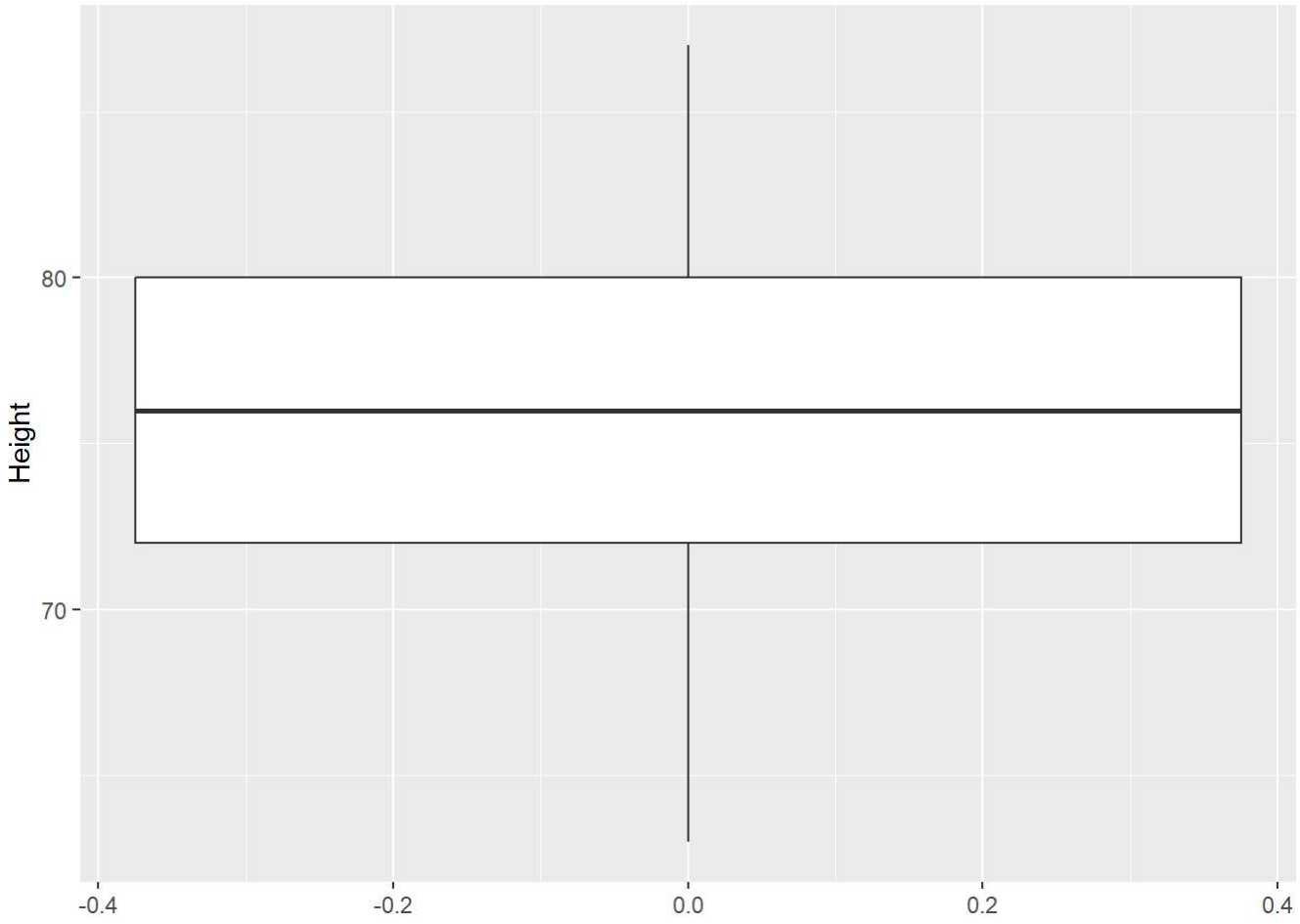


#Question/Action 6

Create a boxplot of the tree heights

Remove the scale_y_continous() portion of the code, you probably won't need it

```
ggplot(trees,aes(y=Height))+geom_boxplot()
```



#Biplots or scatter plots

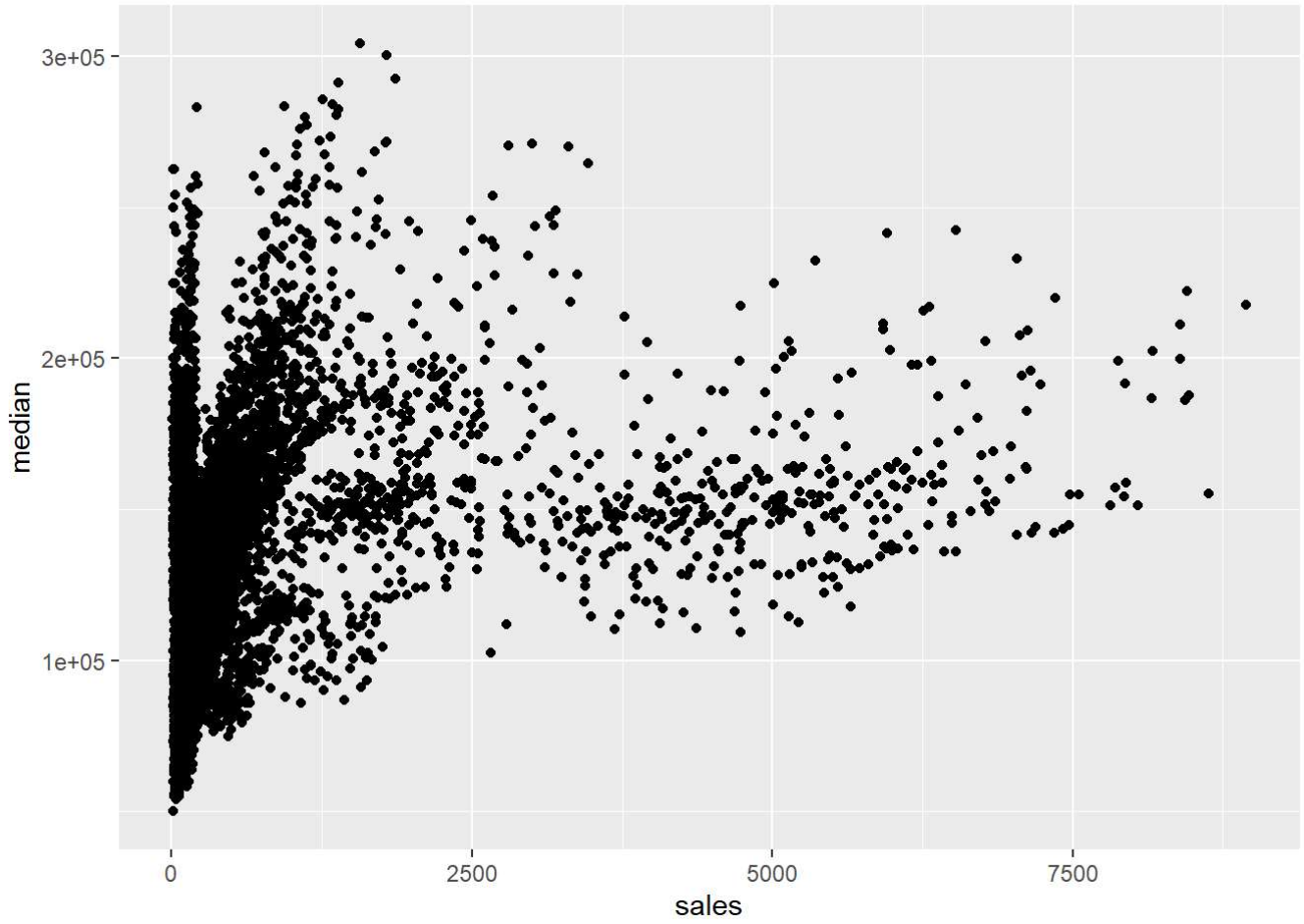
Last one honest!

This is the classic y vs x plot

Let's try median sale price vs number of houses sold

```
ggplot(txhousing,aes(x=sales,y=median))+geom_point()
```

```
## Warning: Removed 617 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



#What does this mean?

The lowest prices are in small or rural counties with few sales.

Larger urban areas have many sales, but few low priced properties

On the other hand, very large plots of land are also in small counties, so the highest medians are also in areas wiht limited numbers of sales.

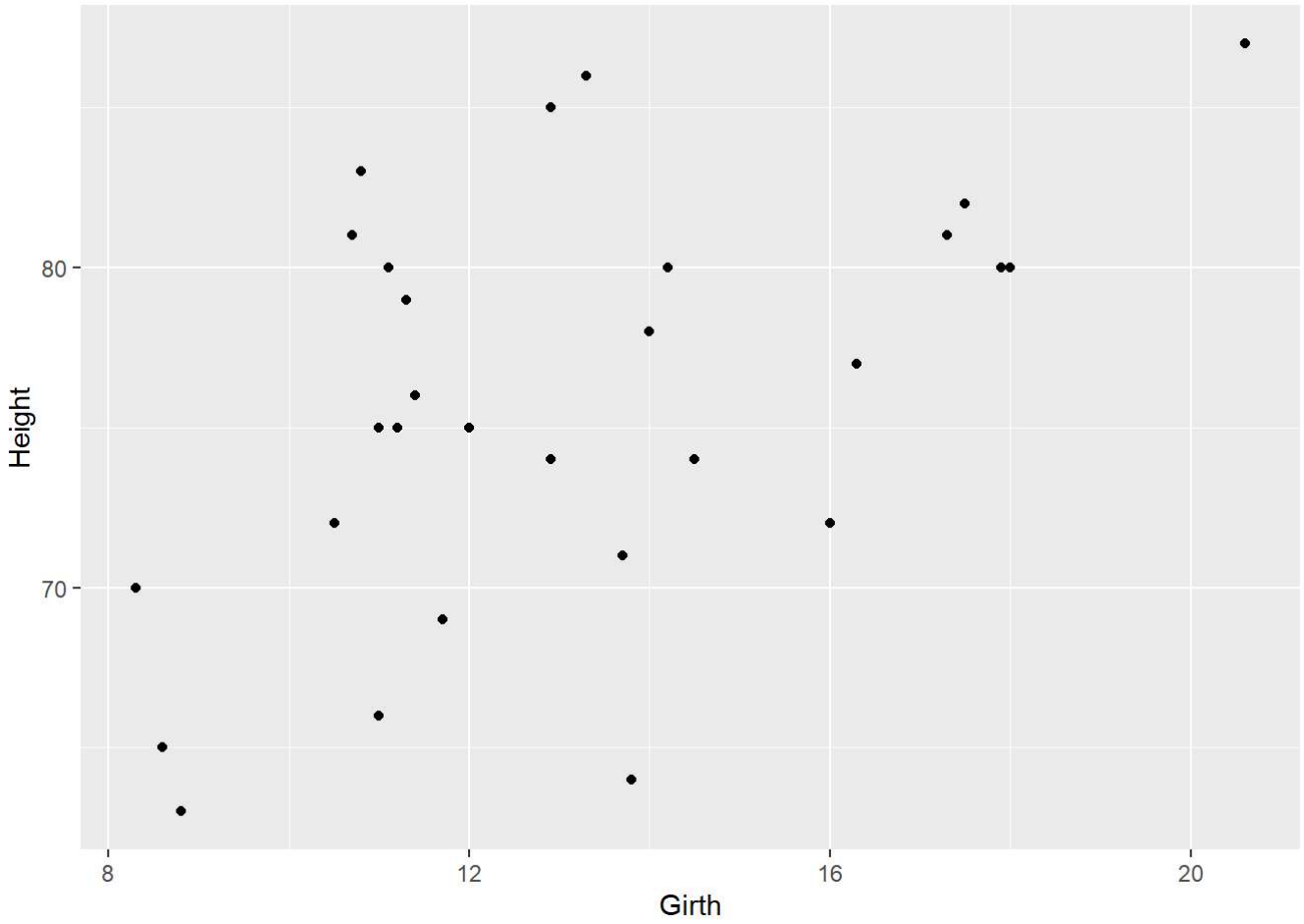
Question/Action 7

Plot tree height (y) vs girth (x)

Do you see what your expect

- Yes, the height grows proportionally with the girth, which could be expected based on the physics of supporting large natural structures.

```
ggplot(trees, aes(x=Girth, y=Height)) + geom_point()
```



#Print this to PDF and Turn it in

-Push the knit button at the top center of the edit window, next to the gear

-Select knit to HTML from the menu

-When it shows you the HTML version, select “Open in Browser”

-In your browser, select the browser print command

-Use the “Print to PDF” option in your browser to create the pdf of this file. Upload it to Canvas