STAT 327 Homework 4

We'll grade your homework by opening your "hw4.Rmd" file in RStudio (in a directory containing "farm.csv", "scores.csv", and "hw4freeCode.R"), clicking "Knit HTML", reading the HTML output, and reading your "hw4.Rmd" file. You should write R code anywhere you see an empty R code chunk.

Name: Naiqing Cai

Email: ncai5@wisc.edu (mailto:ncai5@wisc.edu)

Part 1: A "jackknife" procedure to find the most outlying point in a linear relationship between two variables

First load the "XML" package to give access to `readHTMLTable()` and the "RCurl" package for access to `getURL()`.

```
if (!require("XML")) {
  install.packages("XML") # do this once per lifetime
  stopifnot(require("XML")) # do this once per session
}
```

```
## Loading required package: XML
```

```
if (!require("RCurl")) {
  install.packages("RCurl") # do this once per lifetime
  stopifnot(require("RCurl")) # do this once per session
}
```

```
## Loading required package: RCurl
```

```
## Loading required package: bitops
```

Use R to get the land area (sq. miles) of each state from the second table (labeled "Land area") in the web page https://simple.wikipedia.org/wiki/List_of_U.S._states_by_area (https://simple.wikipedia.org/wiki/List_of_U.S._states_by_area). Hint: you can use `tables = readHTMLTable(getURL("https://simple.wikipedia.org/wiki/List_of_U.S._states_by_area"))` to read the data. Include code to remove the commas from the numbers.

```
tables = readHTMLTable(getURL("https://simple.wikipedia.org/wiki/List_of_U.S._states_by_
area"),header = T)[2]$'NULL'[,c(2,3,4)]
tables[,2]=as.character(tables[,2])
tables[,3]=as.character(tables[,3])
tables[,2]=gsub(pattern = ",","",tables[,2])
tables[,3]=gsub(pattern = ",","",tables[,3])
```

Use R to get farm areas of states from "farm.csv". (Note: I got the data in 2013 from the spreadsheet in the row labeled "825 - Farms–Number and Acreage by State [Excel 131k] …" at http://www.census.gov/compendia/statab/cats/agriculture/farms_and_farmland.html

(http://www.census.gov/compendia/statab/cats/agriculture/farms_and_farmland.html). I took the 2010 acreage (1000) column, multiplied by 1000, and divided by 640 (sq. miles per acre). You do not need to use this spreadsheet–just use “farm.csv”.)

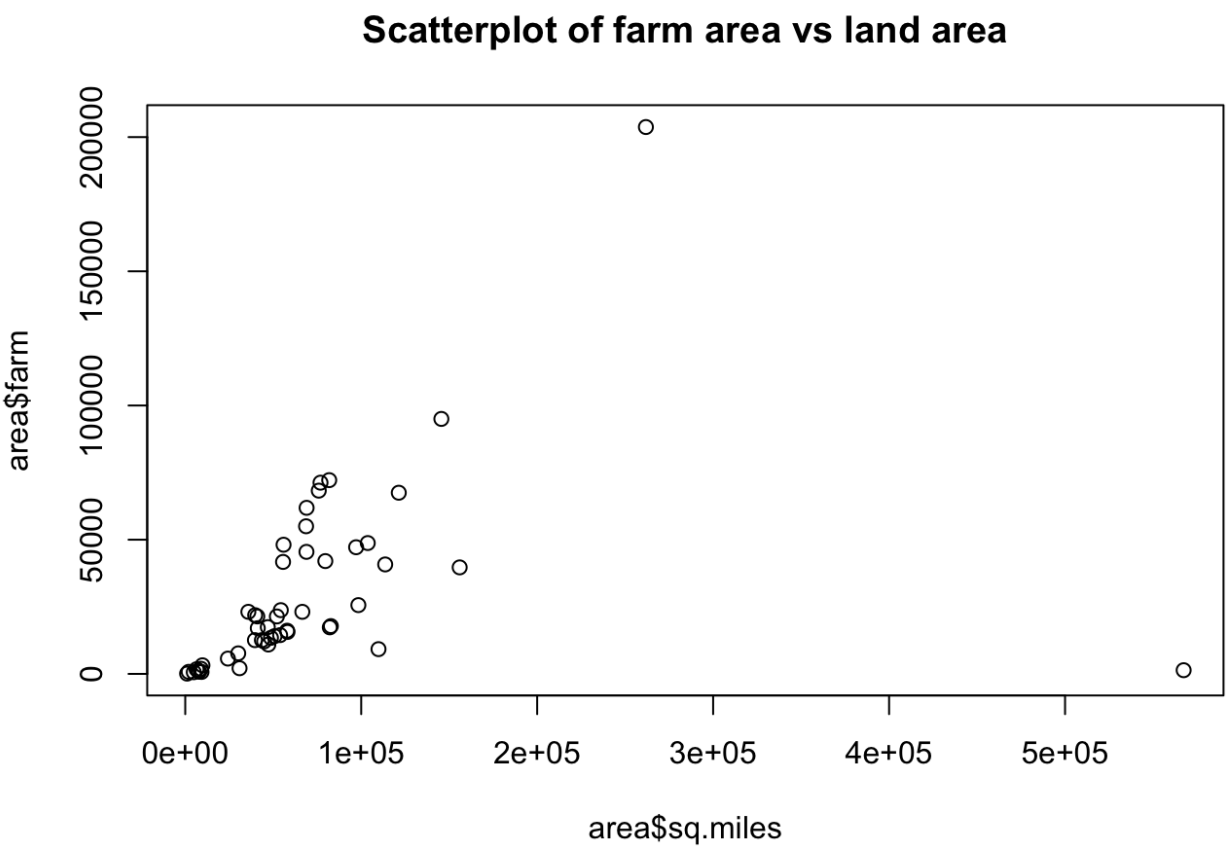
```
farm <- read.csv("farm.csv")
```

Create a data frame called “area” whose columns are “state”, “farm”, and “land”, which contain state names, farm areas, and land areas, respectively. Hint: the states aren’t in the same order in the two data sets, so getting the “area” data frame right requires a little care.

```
farm$state=as.character(farm$state)
farm=farm[order(farm$state),]
tables$State=as.character(tables$State)
tables=tables[1:50,]
tables=tables[order(tables$State),]
area=cbind(tables,farm$sq.miles)
colnames(area)=c("State","km2","sq.miles","farm")
```

Make a scatterplot of y = farm area vs. x = land area.

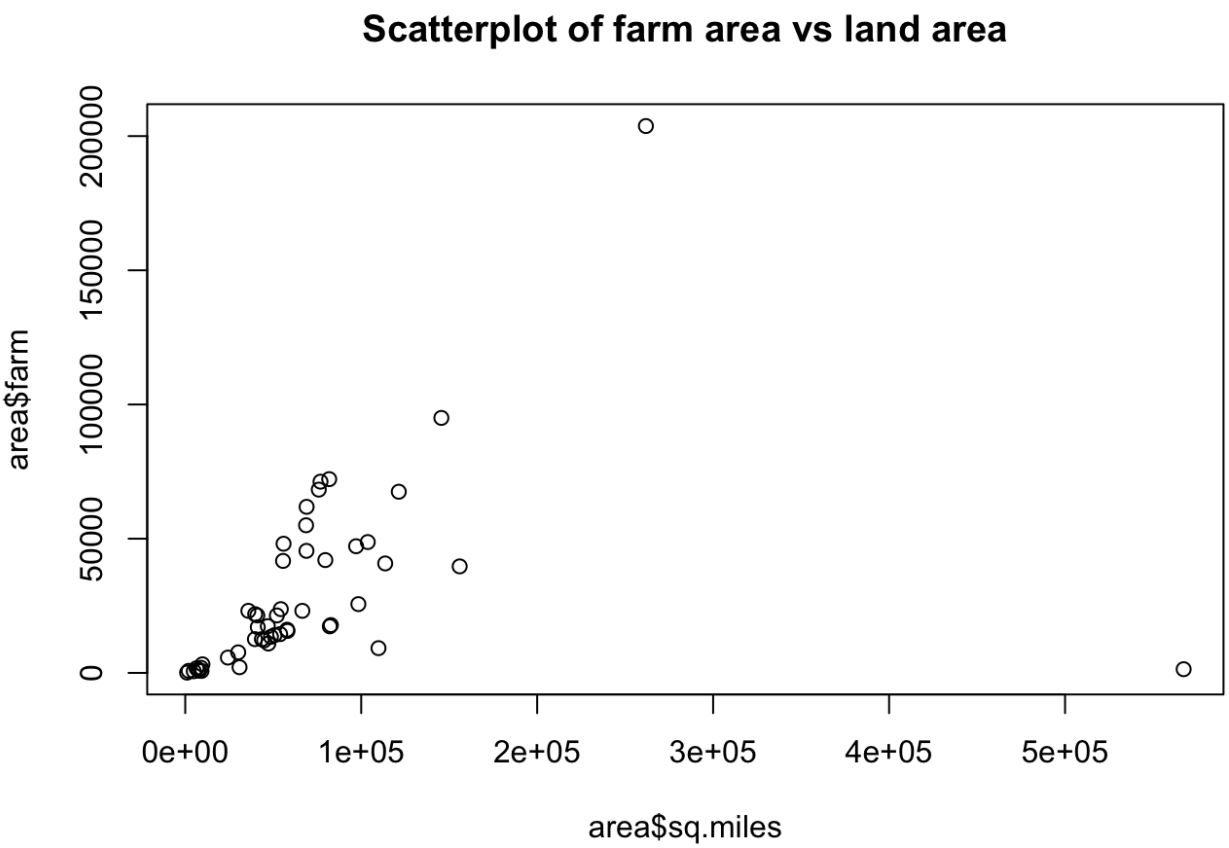
```
plot(area$sq.miles, area$farm,main = "Scatterplot of farm area vs land area")
```



There are two prominent outliers. Use `identify()` to find their indices.

Unfortunately, `identify()` doesn’t work on an R graph that we’re viewing through an HTML page. We can use the RStudio menu command “Chunks > Run all” to run all the code in this file in the console, and then click on the graph in RStudio’s “Plots” tab. Once you know the indices, just assign them to variables so you can use them later. Then comment out your call to identify().

```
plot(area$sq.miles, area$farm,main = "Scatterplot of farm area vs land area")
identify(area$sq.miles, area$farm,n=2)
```

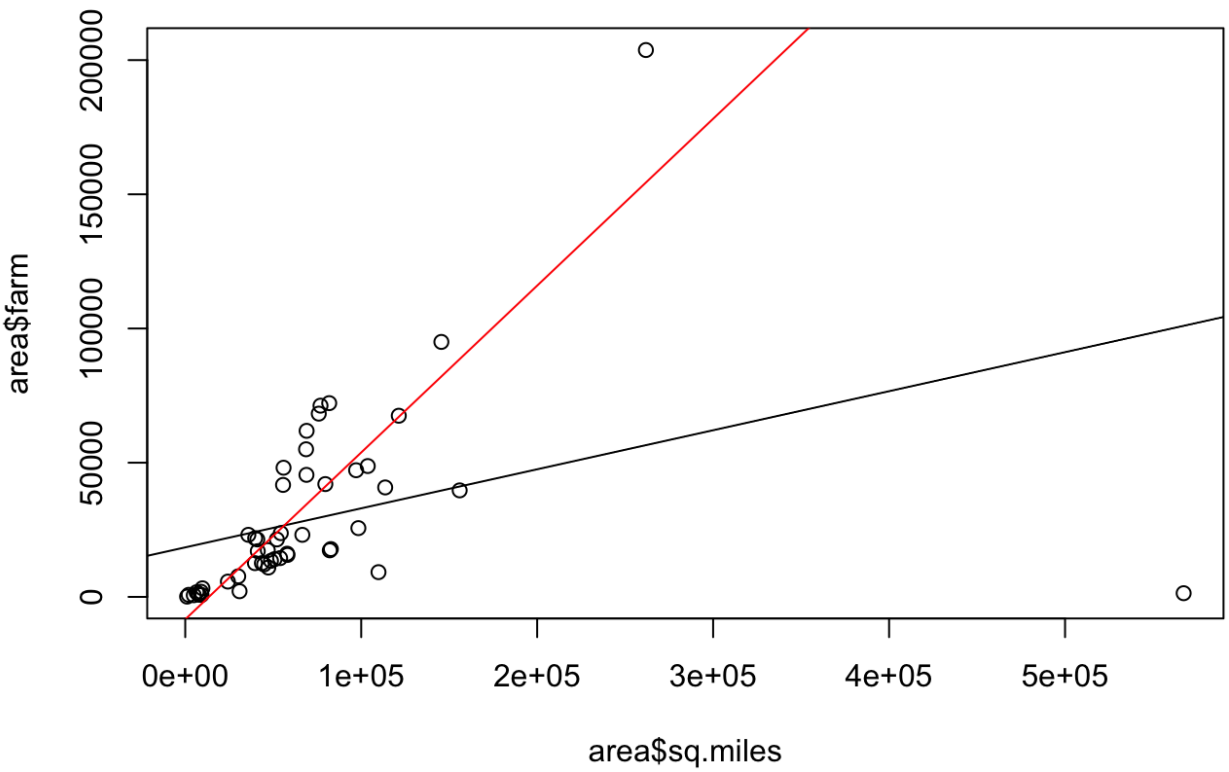



```
## integer(0)
```

The two outliers are Texas, which fits the roughly linear trend of the rest of the data, and Alaska, which does not fit.

Make a linear model of y = farm area vs. x = land area. Make your scatterplot again, and this time add the regression line to it. Then make a linear model of the same data, except with Alaska removed. Add that regression line, colored red, to your scatterplot.

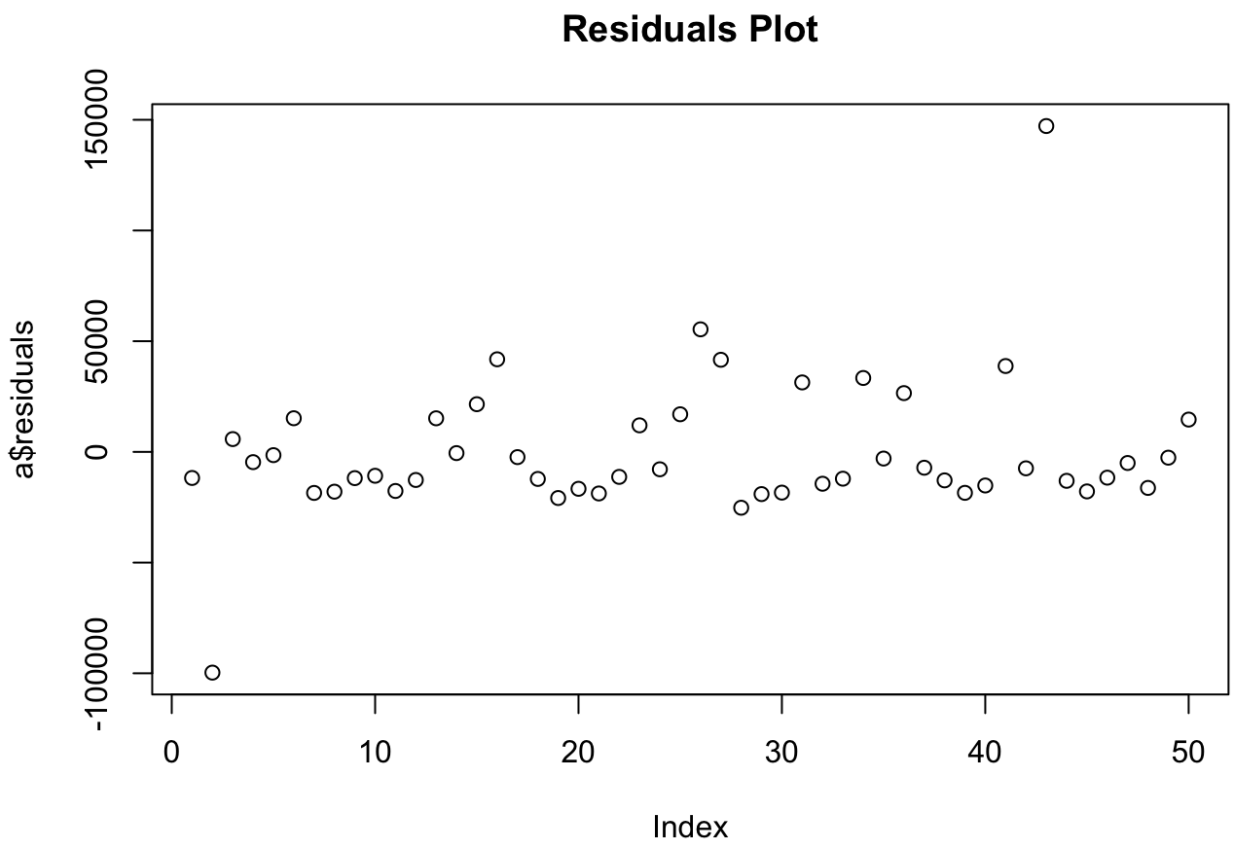
```
plot(area$sq.miles, area$farm)
a=lm(as.numeric(area$farm)~as.numeric(area$sq.miles))
abline(a)
b <- lm(as.numeric(area[-2,]$farm)~as.numeric(area[-2,]$sq.miles))
abline(b, col = "red")
```



Notice that, with respect to the original regression line, Texas has the biggest residual (difference in actual and predicted y), because Alaska pulled the line down toward itself. But really Alaska is the outlier! Next we'll do a "jackknife" procedure to discover computationally that Alaska is the most important outlier.

Make a plot of the residuals for the original model. (Hint: they're available in the output of `lm()`.)

```
plot(a$residuals,main = "Residuals Plot")
```



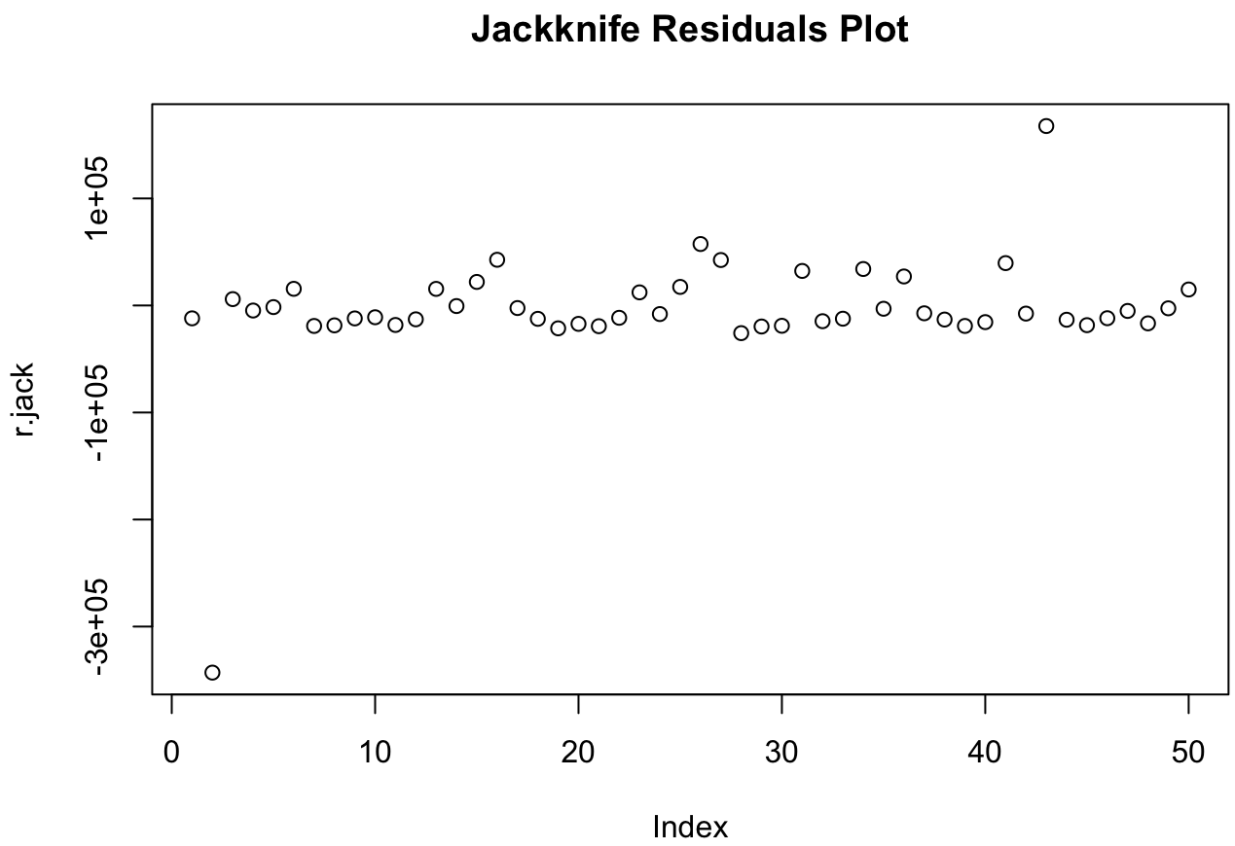
Notice again that the Texas residual is bigger than the Alaska residual.

Next use a loop to create n=50 models. In step i, make a model of the data with observation i removed. Then predict the value of y[i] from that model, and find the residual (difference) between (the removed) y[i] and the prediction. Save these residuals in a vector `r.jack`. (A "jackknife" procedure works by removing one observation (or several) from a data set, and then making a prediction from that smaller data set, and repeating this for each observation.)

```
y=c()
r.jack =c()
for (i in 1:50) {
  model=lm(as.numeric(area[-i,]$farm)~as.numeric(area[-i,]$sq.miles))
  d=as.numeric(area[i,]$sq.miles)
  y[i]=model$coefficients[1]+model$coefficients[2]*d
  r.jack[i]=area$farm[i]-y[i]
}
```

Plot these "jackknife" residuals.

```
plot(r.jack,main = "Jackknife Residuals Plot")
```



Notice now that Alaska is clearly the real outlier.

Part 2: ggplot2 graphics

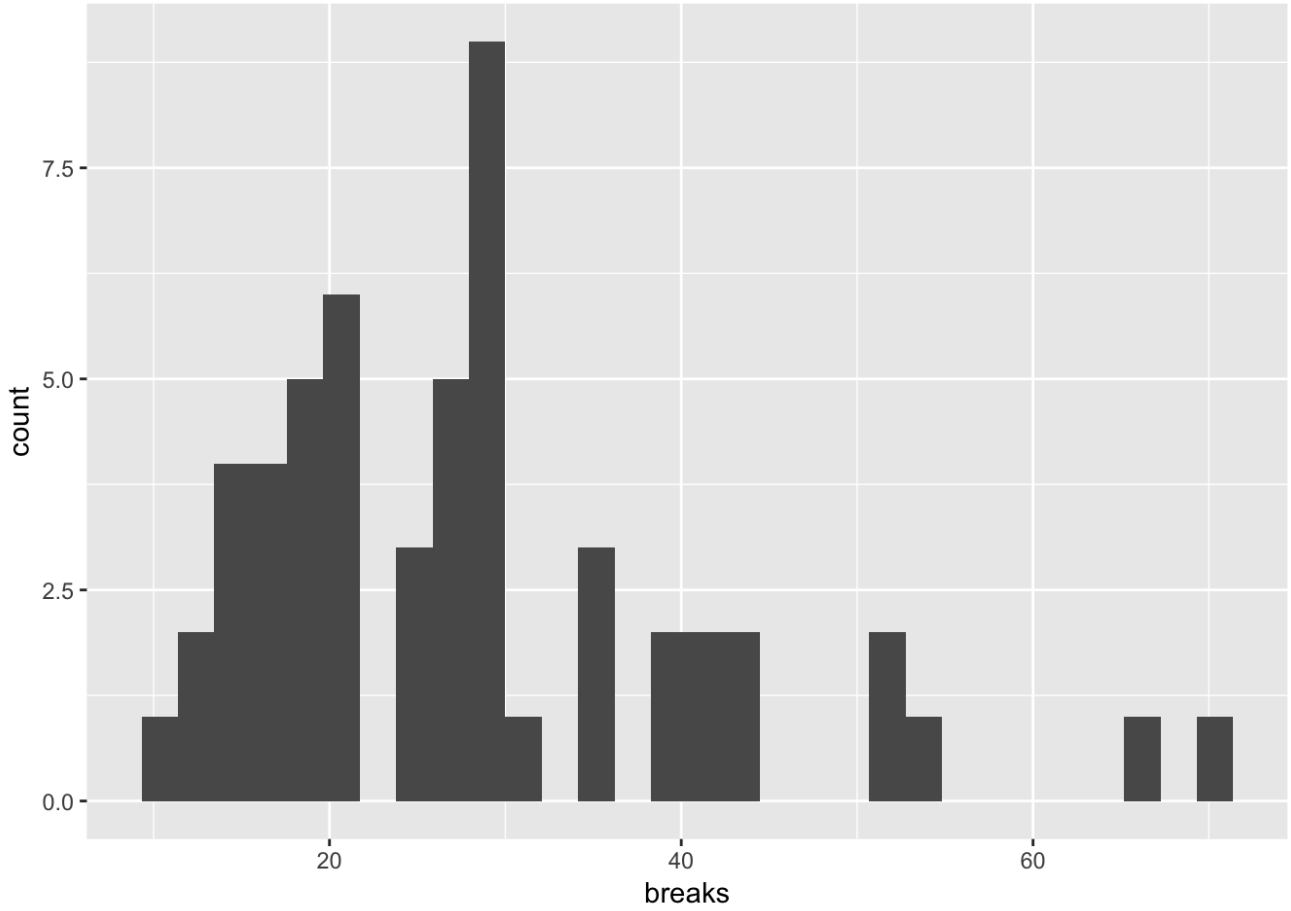
Use `ggplot2` to solve make several graphs. First, here's code to load, or install and load, the package.

```
if (!require("ggplot2")) {
  install.packages("ggplot2")
  stopifnot(require("ggplot2"))
  }
```

```
## Loading required package: ggplot2
```

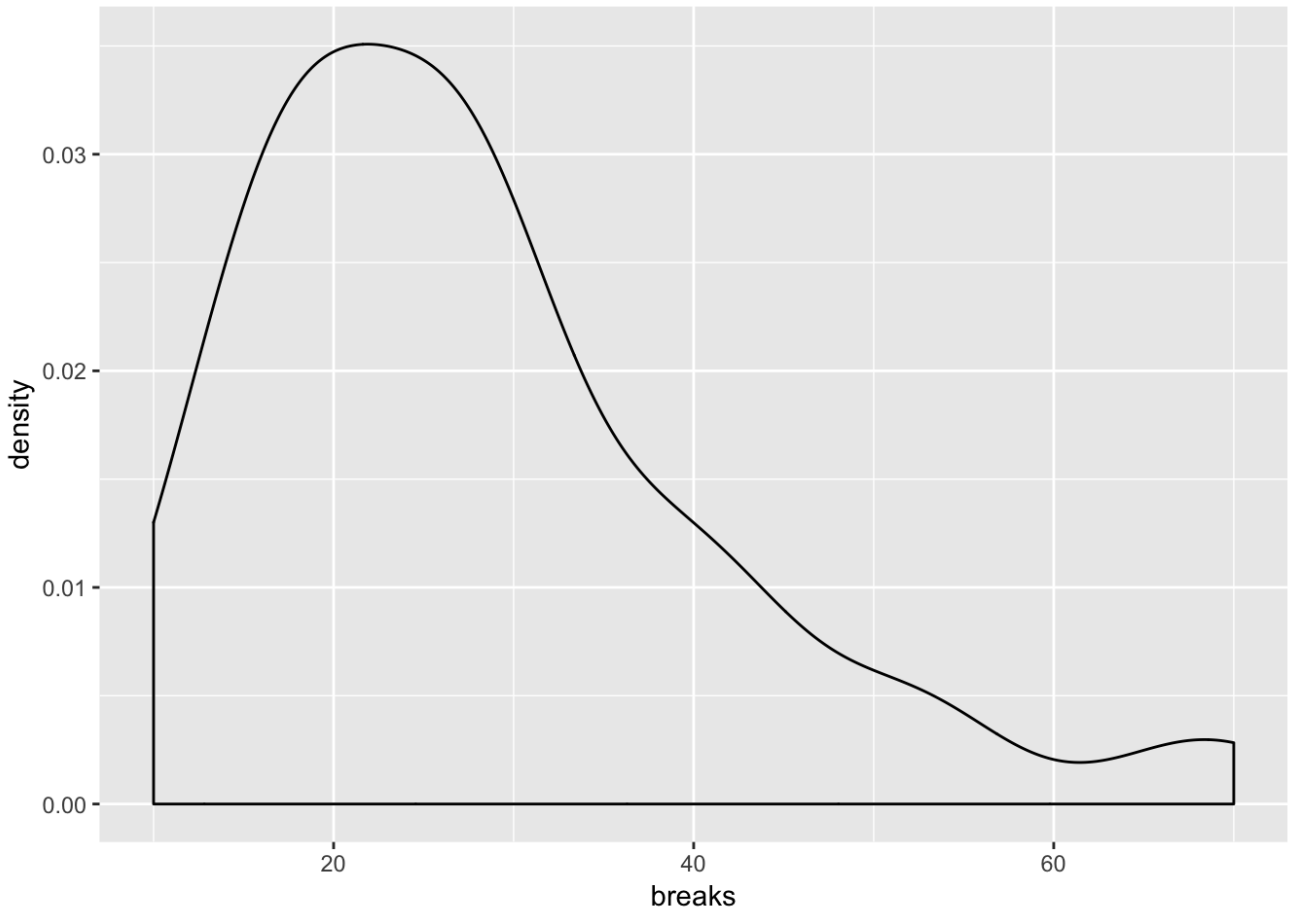
1. Consider the built-in data set `warpbreaks`. (See `?warpbreaks`, http://en.wikipedia.org/wiki/Warp_%28weaving%29 (http://en.wikipedia.org/wiki/Warp_%28weaving%29), and http://en.wikipedia.org/wiki/Power_loom#Operation (http://en.wikipedia.org/wiki/Power_loom#Operation).) Make a histogram of the numbers of warp breaks.

```
warpbreaks <- warpbreaks
ggplot(warpbreaks, aes(x = breaks)) + geom_histogram(bins = 30)
```



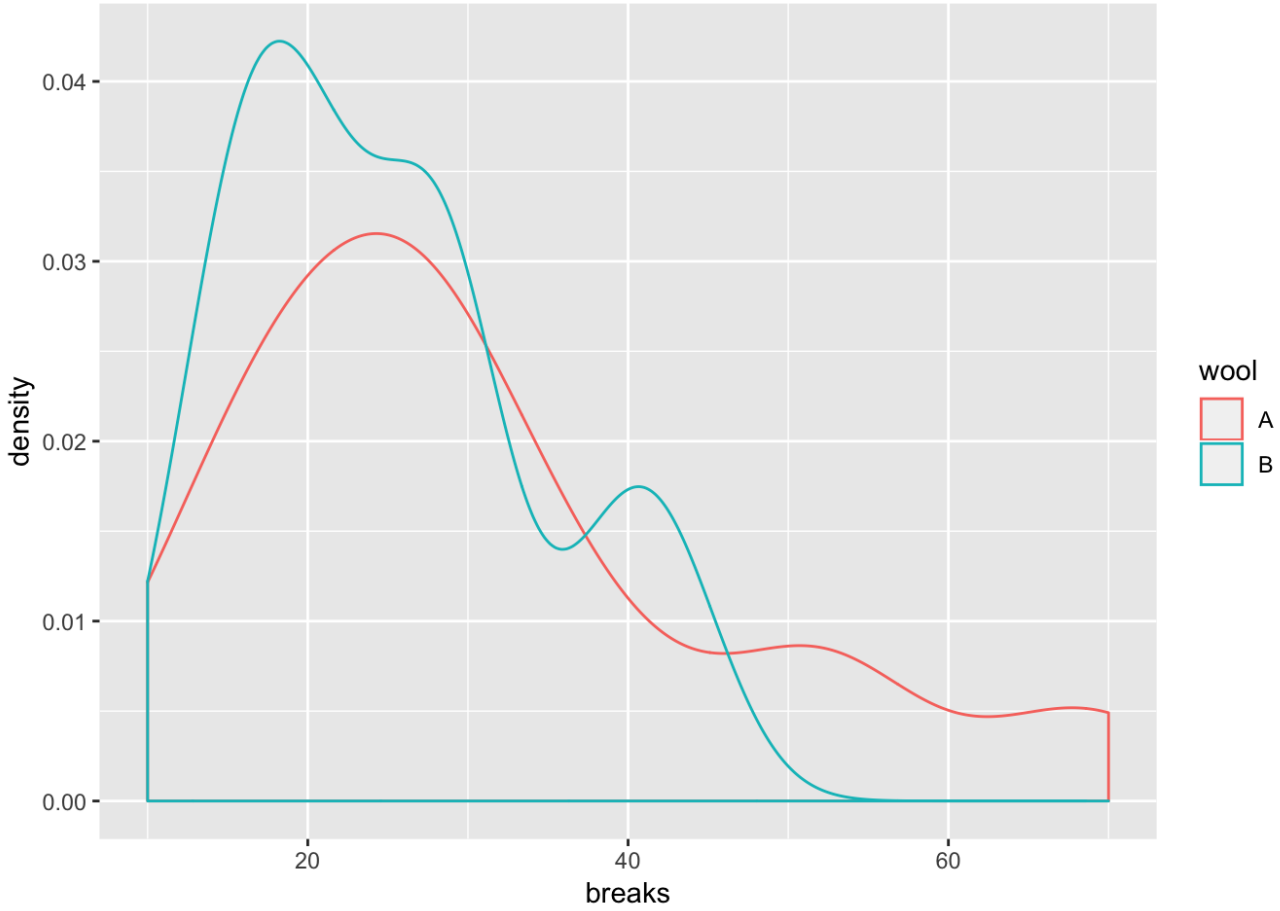
2. Make a density plot of the numbers of warp breaks.

```
ggplot(warpbreaks, aes(x = breaks)) + geom_density()
```



3. Make two density plots of warp breaks, using a different color for each wool type, on a single panel. Does the wool type have a strong effect on the number of breaks?

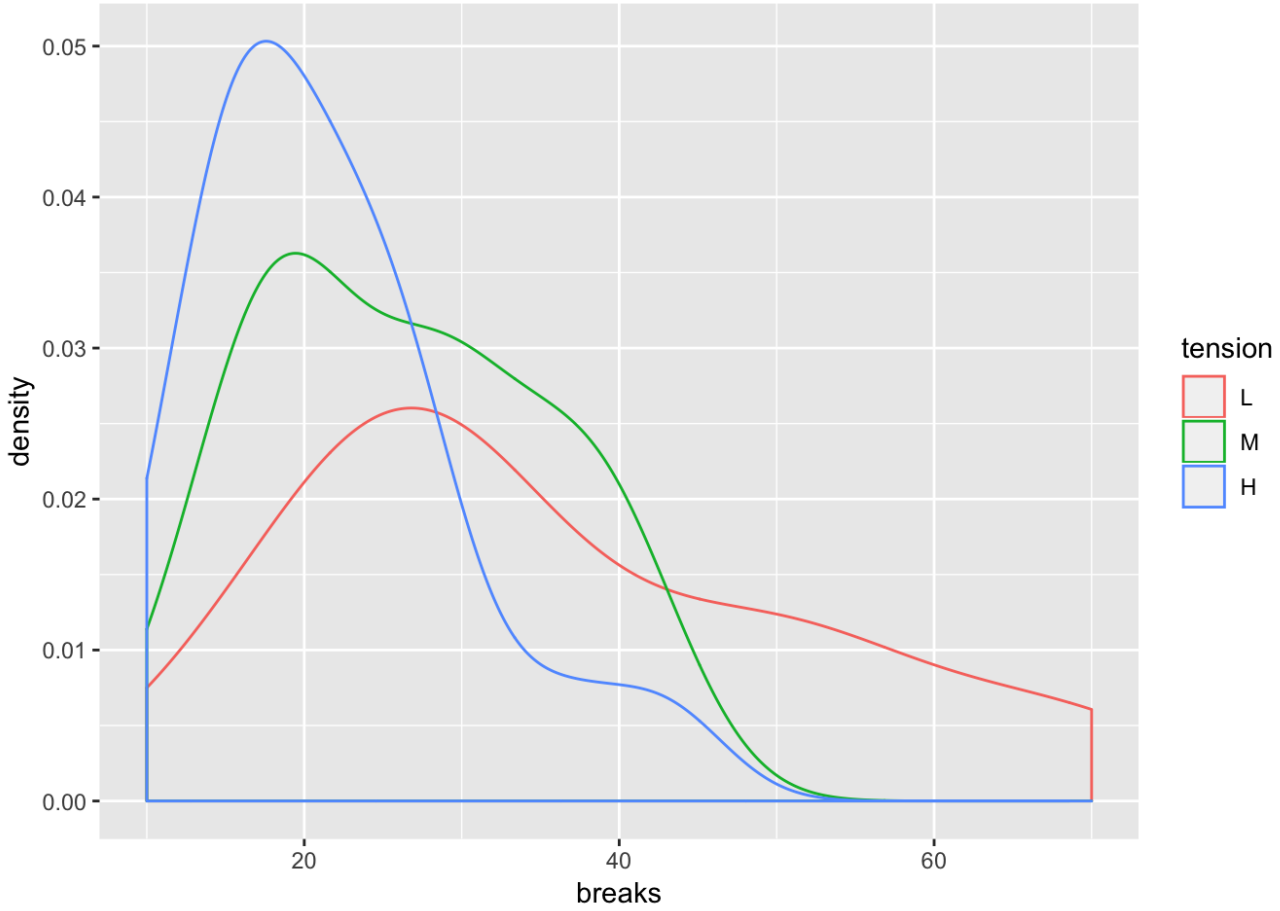
```
ggplot(warpbreaks, aes(x = breaks)) + geom_density(aes(colour = wool))
```



It seems that the wool type have an effect on the number of breaks, type A woll seems to have less breaks.

4. Make three density plots of warp breaks, using a different color for each tension level, on a single panel. How does tension seem to affect the number of breaks?

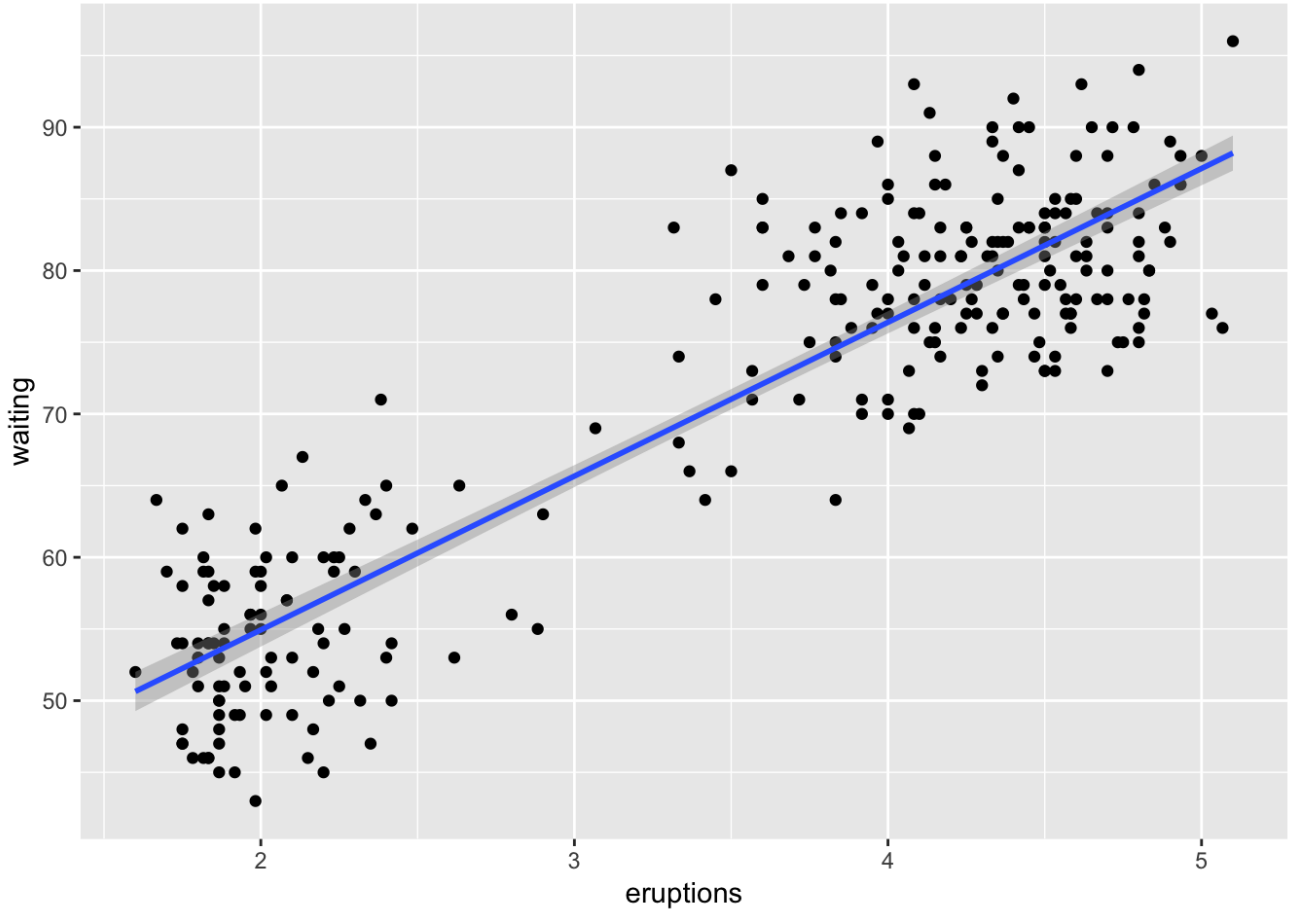
```
ggplot(warpbreaks, aes(x = breaks)) + geom_density(aes(colour = tension))
```



Yes, it has an effect. It seems higher tension leads to less breaks.

5. "Old Faithful" is a geyser in Yellowstone National Park that erupts on a remarkably regular schedule (http://en.wikipedia.org/wiki/Old_Faithful (http://en.wikipedia.org/wiki/Old_Faithful)). Make a scatterplot of waiting time (y) vs. most recent eruption time (x) from the built-in `faithful` data set. (See `?faithful`.) Include a simple linear regression line. What is the most striking feature of this plot?

```
faithful <- faithful
ggplot(faithful, aes(x = eruptions, y = waiting)) + geom_point() + geom_smooth(method =
lm)
```



The most striking feature is that there seem to be no extreme outliers. That is, the length of eruptions and the time between eruptions seems to be very strongly correlated.

Part 3: Web-scraping

```
rm(list=ls())
```

First load the "XML" package to give access to `readHTMLTable()`.

```
if (!require("XML")) {
  install.packages("XML") # do this once per lifetime
  require("XML") # do this once per session
  }
```

At the bottom of the Internet Movie Database website (http://www.imdb.com) there's a link to the Top 250 (http://www.imdb.com/chart/top). At the "Top 250" page there's a list of 250 movies, with a link to each movie. The first movie is The Shawshank Redmption (http://www.imdb.com/title/tt0111161/?ref_=chttp_tt_1).

With your browser on the "Top 250" page, you can do "right-click > view page source" (in Firefox or Chrome; in Safari, first do "Safari > Preferences > Advanced" and check "Show Develop menu in menu bar") to see the HTML code that creates the page. (You do not need to learn HTML for this homework.)

Search in the HTML source page for "Shawshank", and you'll see that it occurs on line 833. Search for "Godfather", and you'll see that it occurs twice, on line 873 for "The Godfather" and on line 913 for "The Godfather: Part II". For each of these three lines, the preceding line contains a link, relative to the main IMDB URL, to that movie's page. Use grep() to figure out what small string is common to the 250 lines, like these three, that contain links to the top 250 movies.

Notice that line 833 for “The Shawshank Redemption” includes the text “/title/tt0111161”. Pasting this onto “http://www.imdb.com (http://www.imdb.com)” gives “http://www.imdb.com/title/tt0111161 (http://www.imdb.com/title/tt0111161)”, which is a link to the first movie’s page. Adding “/fullcredits” gives “http://www.imdb.com/title/tt0111161/fullcredits (http://www.imdb.com/title/tt0111161/fullcredits)”, which is a link to the full cast and crew. Search this “fullcredits” page for “Produced” and you’ll see that “The Shawshank Redemption” was produced by “Liz Glotter”, “David V. Lester”, and “Niki Marvin”.

Write code that does the following:

- Use readLines() to read “http://www.imdb.com/chart/top (http://www.imdb.com/chart/top)” into a character string vector
    - Select the 250 lines containing links to the 250 movies
    - From these 250 lines, select the 250 strings like “/title/tt0111161” from which you can form links to the 250 movies
- Create an empty list of producers, e.g. “producers = list()”
- Read the “fullcredits” page of each movie
    - Strip out the title of the movie
    - Use readHTMLTable() to read all the tables into a list of dataframes; figure out which dataframe has the producers
    - Save the vector of producers in a list, doing something like “producers[[title]] = …”, where “…” is the vector of producers you found
- Do “unlist(producers)” to convert your list of title / producer vector pairs into a named vector of producers.
    - Use table() to make a table of counts from this vector
    - Display the 5 producers who produced the most movies from among these 250

```
a=readLines("https://www.imdb.com/chart/top")
b=a[grep("   <a href=\"/title/tt.*chttp_tt_\\d+\"",a)]
producers = list()
c=gsub("   <a href=\"(/title/tt\\d+).*chttp_tt_\\d+\"","https://www.imdb.com\\1/fullcred
its",b)
c=gsub("   (.*)","\\1",c)
title=c()
d=c()
for (i in 1:length(c)){
  tables<-readHTMLTable(getURL(c[i]),header = F)
  d[[i]]=readLines(c[i])
  e=grep("    <meta property='og:title' content=\".*\" />",d[[i]])
  title[i]=gsub("    <meta property='og:title' content=\"(.*)\" />","\\1",d[[i]][e])
  producers[[i]]=as.character(tables[[4]]$V1)
}
names(producers)=title
uproducers=unlist(producers)
t=table(uproducers)
sort(t,decreasing = T)[1:5]
```

```
## uproducers
##    Bob Weinstein Harvey Weinstein    John Lasseter    Arnon Milchan
##                9                9                9                7
##      Emma Thomas
##                7
```

Part 4: Extra Credit (not required; worth 0, 1, or 2 points)

- Collect Year, Director, Rating, Number of Votes and Cast (first billed only)

- For each actor, count how many times he or she starred in a Top 250 Movie. Show the 10 actors/actresses that starred in the most movies among the Top 250. Show the 10 actors/actresses that starred in movies among the Top 250 with the highest mean rating.
- For each director, count how many times he or she directed a Top 250 Movie. Show the 10 directors that directed the most movies among the Top 250. Show the 10 directors that directed movies among the Top 250 with the highest mean rating.
- Show the 10 most frequent Actor-Director collaborations among the Top 250 Movies. What's the average rating for those collaborations?
- Are ratings influenced by year? In what way? Provide a P-value using linear regression. Are the assumptions of linear regression violated? If so, what's the impact in your P-value estimate?
- Do people vote more often for recent movies? Provide a P-value using linear regression. Are the assumptions of linear regression violated? If so, what's the impact in your P-value estimate?
- In light of the previous question, do you think the number of votes influences the rating? Create an analysis of variance table for the ratings, considering year, votes and the interaction of year and votes. Explain what the interaction means.

```
year=c()
director=c()
rating=c()
numofvotes=c()
cast=list()
stars=list()
for (i in 1:length(c)){
  year[i]=gsub(".*\\((\\d+)\\)","\\1",title[i])
  director[i]=gsub("> (.*)","\\1",d[[i]][grep("<a href=\"/name/nm\\d+/.*ref_=ttfc_fc_dr1
\"",d[[i]])+1])
}
c1=gsub("   <a href=\"(/title/tt\\d+).*chttp_tt_\\d+\"","http://www.imdb.com\\1",b)
c1=gsub("   (.*)","\\1",c1)
d1=c()
for (i in 1:length(c1)){
    d1[[i]]=readLines(c1[i])
    rating[i]=gsub("    \"ratingValue\": \"(.*)\"","\\1",d1[[i]][grep("  \"aggregateRati
ng\": \\{",d1[[i]])+5])
    numofvotes[i]=gsub("    \"ratingCount\": (.*),","\\1",d1[[i]][grep("  \"aggregateRat
ing\": \\{",d1[[i]])+2])
    cast[[i]]=gsub("> (.*)","\\1",d1[[i]][grep("<a href=\"/name/nm\\d+/.*_=tt_cl_t\\d+
\"",d1[[i]])+1])
    stars[[i]]=c(gsub(">(.*)</a>.*","\\1",d1[[i]][grep("        <h4 class=\"inline\">Sta
rs:</h4>",d1[[i]])+2]),
                 gsub(">(.*)</a>.*","\\1",d1[[i]][grep("        <h4 class=\"inline\">Star
s:</h4>",d1[[i]])+3]),
                 gsub(">(.*)</a>.*","\\1",d1[[i]][grep("        <h4 class=\"inline\">Star
s:</h4>",d1[[i]])+4]))
}

ustars=unlist(stars)
st=table(ustars)
sort(st,decreasing = T)[1:10]
```

```
## ustars
##    Robert De Niro     Harrison Ford Leonardo DiCaprio         Tom Hanks
##                 8                 7                 6                 6
##        Aamir Khan   Charles Chaplin    Clint Eastwood         Al Pacino
##                 5                 5                 5                 4
##         Brad Pitt    Christian Bale
##                 4                 4
```

```
unique(unlist(stars))[1:10]
```

```
##  [1] "Tim Robbins"    "Morgan Freeman" "Bob Gunton"     "Marlon Brando"
##  [5] "Al Pacino"      "James Caan"     "Robert De Niro" "Robert Duvall"
##  [9] "Christian Bale" "Heath Ledger"
```

```
sd=table(director)
sort(sd,decreasing = T)[1:10]
```

```
## director
## Christopher Nolan   Martin Scorsese   Stanley Kubrick  Steven Spielberg
##                 7                 7                 7                 7
##    Akira Kurosawa  Alfred Hitchcock    Hayao Miyazaki      Billy Wilder
##                 6                 6                 6                 5
##   Charles Chaplin Quentin Tarantino
##                 5                 5
```

```
unique(director)[1:10]
```

```
##  [1] "Frank Darabont"       "Francis Ford Coppola" "Christopher Nolan"
##  [4] "Sidney Lumet"         "Steven Spielberg"     "Peter Jackson"
##  [7] "Quentin Tarantino"    "Sergio Leone"         "David Fincher"
## [10] "Robert Zemeckis"
```

```
names(stars)=director
ds=unlist(stars)
name=names(ds)
name=gsub("(.*)\\d","\\1",name)
names(ds)=1:length(ds)
dslist=list()
for (i in 1:length(ds)){
  dslist[[i]]=paste(name[i],",",ds[i])
}
colt=table(unlist(dslist))
sort(colt,decreasing = T)[1:10]
```

```
##
##   Charles Chaplin , Charles Chaplin     Akira Kurosawa , Toshirô Mifune
##                                   5                                    4
##  Christopher Nolan , Christian Bale    Martin Scorsese , Robert De Niro
##                                   4                                    4
##    Akira Kurosawa , Tatsuya Nakadai     Clint Eastwood , Clint Eastwood
##                                   3                                    3
##         Martin Scorsese , Joe Pesci Martin Scorsese , Leonardo DiCaprio
##                                   3                                    3
##         Peter Jackson , Elijah Wood        Peter Jackson , Ian McKellen
##                                   3                                    3
```

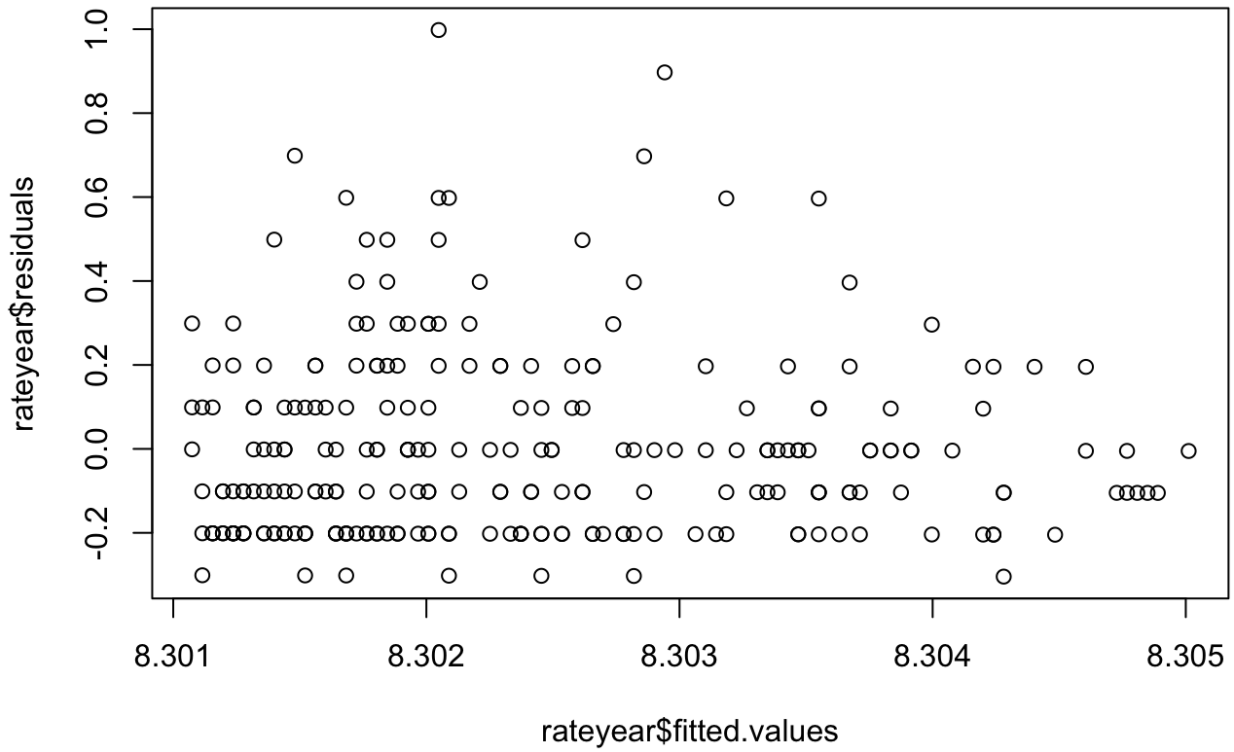
```
dironlist=gsub("(.*) ,.*","\\1",names(sort(colt,decreasing = T)[1:10]))
names(stars)=director
aaa=list()
for (i in 1:length(dironlist)){
  aaa[[i]]=grep(names(sort(colt,decreasing = T)[1:10])[i],unlist(dslist))
}
averrate=c()
for (i in 1:10){
  averrate[i]=mean(as.numeric(rating[sort(aaa[[i]]%/%3)]))
}
names(averrate)=names(sort(colt,decreasing = T)[1:10])

year=as.numeric(year)
rating=as.numeric(rating)
rateyear=lm(rating~year)
summary(rateyear)
```

```
##
## Call:
## lm(formula = rating ~ year)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.30428 -0.20139 -0.10116  0.09851  0.99795
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.383e+00  1.158e+00   7.238 5.68e-12 ***
## year        -4.059e-05  5.833e-04  -0.070    0.945
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2281 on 248 degrees of freedom
## Multiple R-squared:  1.952e-05,  Adjusted R-squared:  -0.004013
## F-statistic: 0.004841 on 1 and 248 DF,  p-value: 0.9446
```

p-value=0.945. Ratings will not be influenced by year.

```
plot(rateyear$fitted.values,rateyear$residuals)
```



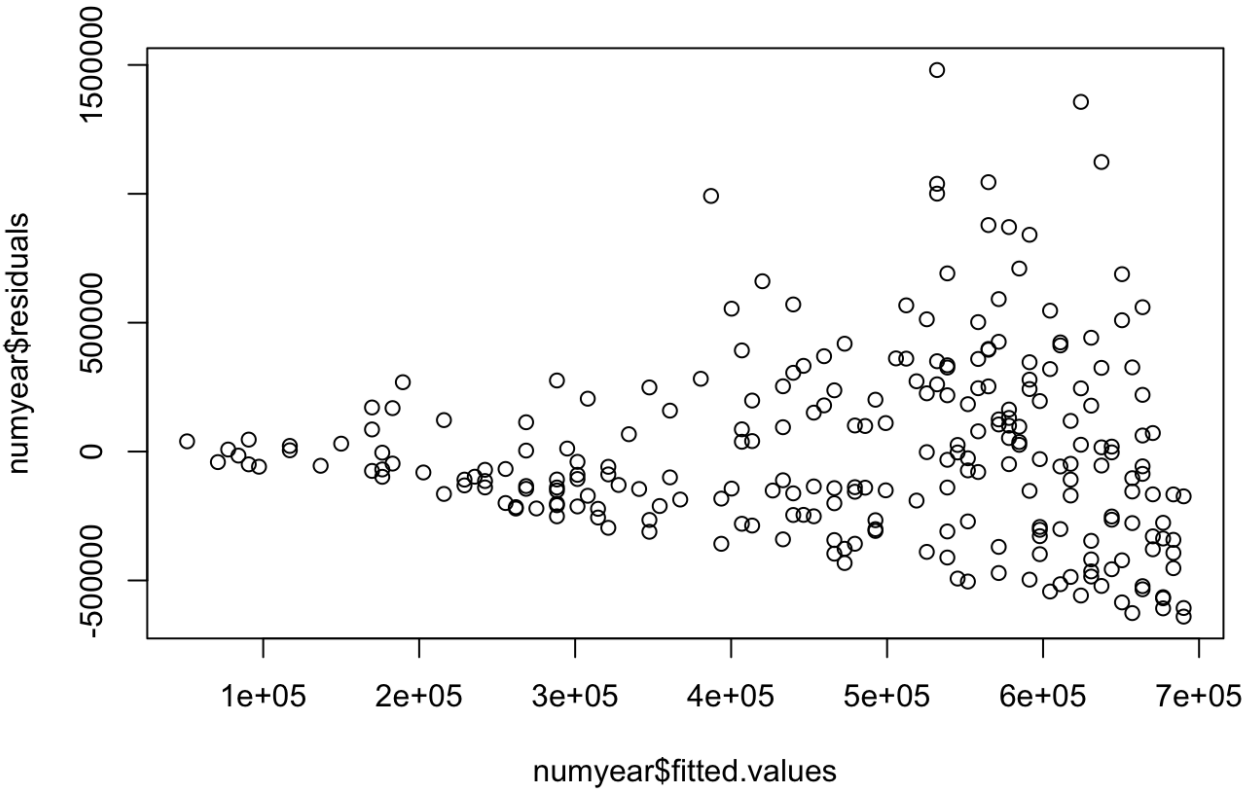
The assumptions of linear regression are violated. The assumption equal variance is violated.

```
numofvotes=as.numeric(numofvotes)
numyear=lm(numofvotes~year)
summary(numyear)
```

```
##
## Call:
## lm(formula = numofvotes ~ year)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -640284 -245930  -58355  197407 1480219
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.260e+07  1.850e+06  -6.812 7.26e-11 ***
## year         6.587e+03  9.318e+02   7.069 1.58e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 364300 on 248 degrees of freedom
## Multiple R-squared:  0.1677, Adjusted R-squared:  0.1643
## F-statistic: 49.97 on 1 and 248 DF,  p-value: 1.58e-11
```

p-value=1.58e-11 and so people vote more often for recent movies.

```
plot(numyear$fitted.values,numyear$residuals)
```



The assumptions of linear regression are violated. The assumption equal variance is violated.

```
ratyearvote=aov(rating~year*numofvotes)
anova(ratyearvote)
```

```
## Analysis of Variance Table
##
## Response: rating
##                  Df Sum Sq Mean Sq  F value    Pr(>F)
## year              1 0.0003  0.0003   0.0091     0.924
## numofvotes        1 5.2917  5.2917 191.7172 < 2.2e-16 ***
## year:numofvotes   1 0.8167  0.8167  29.5890 1.288e-07 ***
## Residuals       246 6.7899  0.0276
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The number of votes influences the rating. The interaction means Year and number of votes are linearly related