Exercise 8 of Section 2.4

(a) To begin the exercise we start by reading the college data set into R from the college.csv file. We use the read.csv() function for this and set the data table equal to college, making sure to set the correct directory.

```
college = read.csv('/Users/stevenfutter/Dropbox/NU/MACHINE_LEARNING/College.csv')
```

(b) Next we look at the data using the fix() function. fix() returns an output similar to that of an Excel spreadsheet. Since the first column is the names of the universities, we do not want to treat them as data points, but we will store them into a variable as they may be useful to have later in the analysis. Below I add an additional column row.names with the names of the universities.See figure 1 output below.

```
rownames(college) = college[,1]
fix(college)
```

R has now given each row a name corresponding to the appropriate university. We now eliminate the first column of data where the names are stored.

```
college = college[,-1]
fix(college)
```

The fix application produces the sample output provided below. Note that not all columns are included in the output given the width of the fix() returned output extends beyond my screen view, but note that the first column is row.names and the second column is now ‘Private’.

Fig 1

R Data Editor

Copy Paste Quit

	row.names	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board
1	Abilene Christian University	Yes	1660	1232	721	23	52	2885	537	7440	3300
2	Adelphi University	Yes	2186	1924	512	16	29	2683	1227	12280	6450
3	Adrian College	Yes	1428	1097	336	22	50	1036	99	11250	3750
4	Agnes Scott College	Yes	417	349	137	60	89	510	63	12960	5450
5	Alaska Pacific University	Yes	193	146	55	16	44	249	869	7560	4120
6	Albertson College	Yes	587	479	158	38	62	678	41	13500	3335
7	Albertus Magnus College	Yes	353	340	103	17	45	416	230	13290	5720
8	Albion College	Yes	1899	1720	489	37	68	1594	32	13868	4826
9	Albright College	Yes	1038	839	227	30	63	973	306	15595	4400
10	Alderson-Broaddus College	Yes	582	498	172	21	44	799	78	10468	3380

(c) i. Here we use the summary() function to produce a numerical summary of the variables in the college data set. The summary function produces key statistics for the **min, max, 1st & 3rd quartiles, median, and mean** values for each variable in the data frame, producing the output in figure 2. As can be seen the the maximum number of applications received by any college is 48,094, yet the maximum number of new students enrolled is only 6,392.

```
summary(college)
```

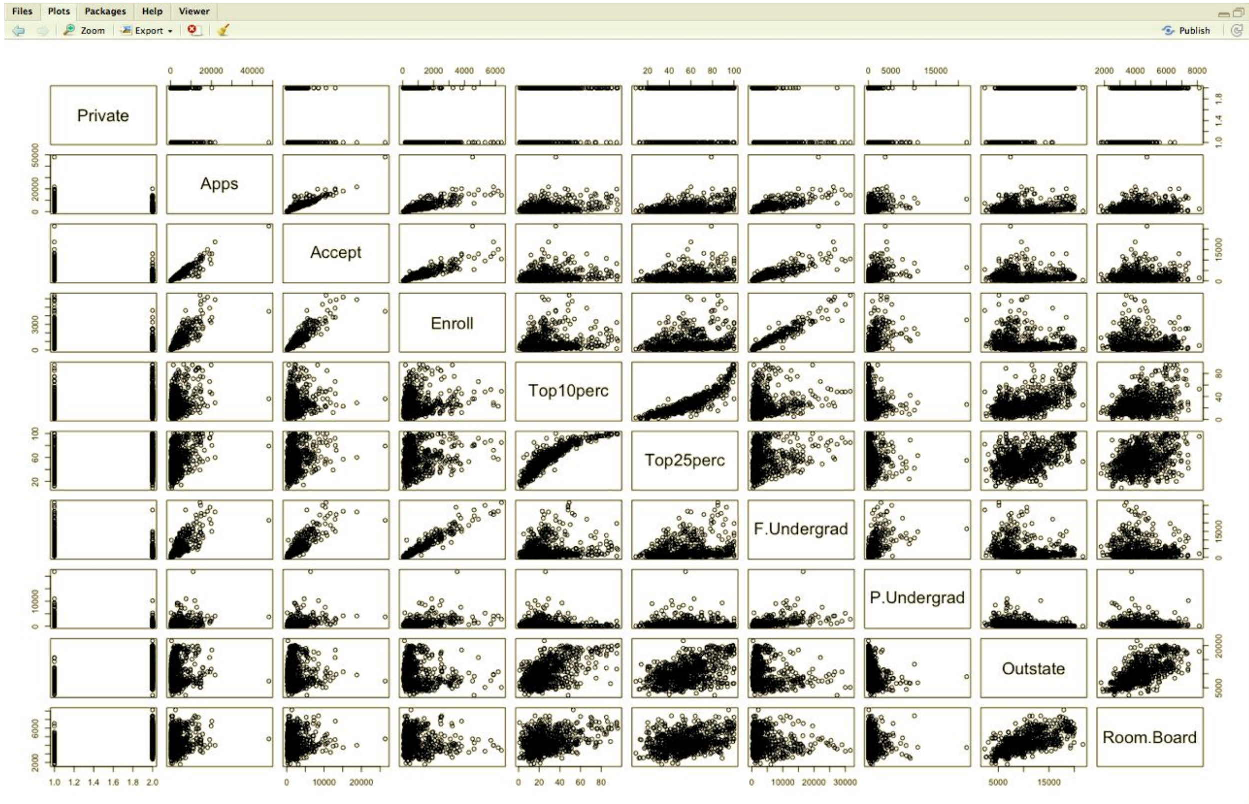
Fig 2

```
> summary(college)
 Private        Apps           Accept          Enroll        Top10perc       Top25perc      F.Undergrad     P.Undergrad         Outstate       Room.Board
 No :212   Min.   :   81   Min.   :   72   Min.   :  35   Min.   : 1.00   Min.   :  9.0   Min.   :  139   Min.   :    1.0   Min.   : 2340   Min.   :1780
 Yes:565   1st Qu.:  776   1st Qu.:  604   1st Qu.: 242   1st Qu.:15.00   1st Qu.: 41.0   1st Qu.:  992   1st Qu.:   95.0   1st Qu.: 7320   1st Qu.:3597
           Median : 1558   Median : 1110   Median : 434   Median :23.00   Median : 54.0   Median : 1707   Median :  353.0   Median : 9990   Median :4200
           Mean   : 3002   Mean   : 2019   Mean   : 780   Mean   :27.56   Mean   : 55.8   Mean   : 3700   Mean   :  855.3   Mean   :10441   Mean   :4358
           3rd Qu.: 3624   3rd Qu.: 2424   3rd Qu.: 902   3rd Qu.:35.00   3rd Qu.: 69.0   3rd Qu.: 4005   3rd Qu.:  967.0   3rd Qu.:12925   3rd Qu.:5050
           Max.   :48094   Max.   :26330   Max.   :6392   Max.   :96.00   Max.   :100.0   Max.   :31643   Max.   :21836.0   Max.   :21700   Max.   :8124
     Books           Personal         PhD            Terminal       S.F.Ratio      perc.alumni        Expend        Grad.Rate
 Min.   :  96.0   Min.   : 250   Min.   :  8.00   Min.   : 24.0   Min.   : 2.50   Min.   : 0.00   Min.   : 3186   Min.   : 10.00
 1st Qu.: 470.0   1st Qu.: 850   1st Qu.: 62.00   1st Qu.: 71.0   1st Qu.:11.50   1st Qu.:13.00   1st Qu.: 6751   1st Qu.: 53.00
 Median : 500.0   Median :1200   Median : 75.00   Median : 82.0   Median :13.60   Median :21.00   Median : 8377   Median : 65.00
 Mean   : 549.4   Mean   :1341   Mean   : 72.66   Mean   : 79.7   Mean   :14.09   Mean   :22.74   Mean   : 9660   Mean   : 65.46
 3rd Qu.: 600.0   3rd Qu.:1700   3rd Qu.: 85.00   3rd Qu.: 92.0   3rd Qu.:16.50   3rd Qu.:31.00   3rd Qu.:10830   3rd Qu.: 78.00
 Max.   :2340.0   Max.   :6800   Max.   :103.00   Max.   :100.0   Max.   :39.80   Max.   :64.00   Max.   :56233   Max.   :118.00
```

(c) ii. By using the pairs() function we next produce a scatterplot matrix of the first ten columns or variables of the data. Note that all rows are included, but only the first 10 columns using the square bracket notation. This produces a matrix of correlation scatterplots, as can be see in figure 3. Many variables show a positive correlation with each other.

```
pairs(college[,1:10])
```

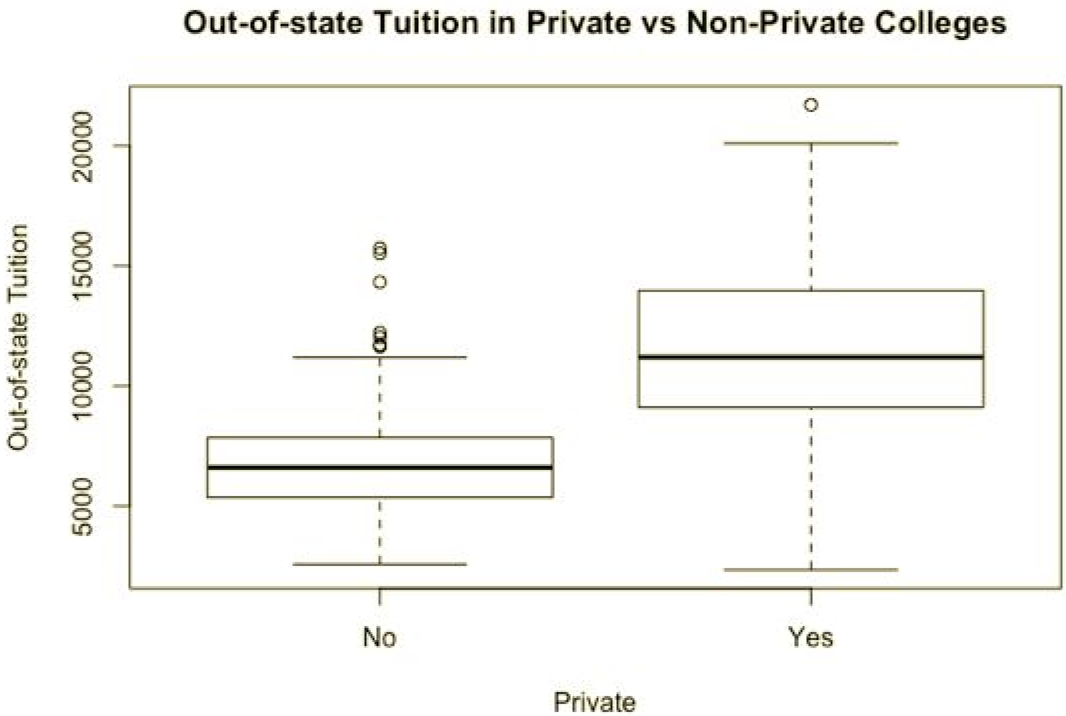
Fig 3



(c) iii. Next we create a boxplot of out-of-state tuition in private versus non-private colleges. Note that we use the boxplot function to create the display in figure 4 below. Not surprisingly, the private college tuition is considerably more expensive. Median costs are almost twice as high in private schools compared to their non-private counterparts. Note the use of the attach() function. This is used so that we do not have to write out the data frame columns using the $ notation each time.

```
attach(college)
boxplot(Outstate~Private,data=college, main="Out-of-state Tuition in Private vs Non-Private Colleges", xlab="Private", ylab="Out-of-state Tuition")
```

Fig 4

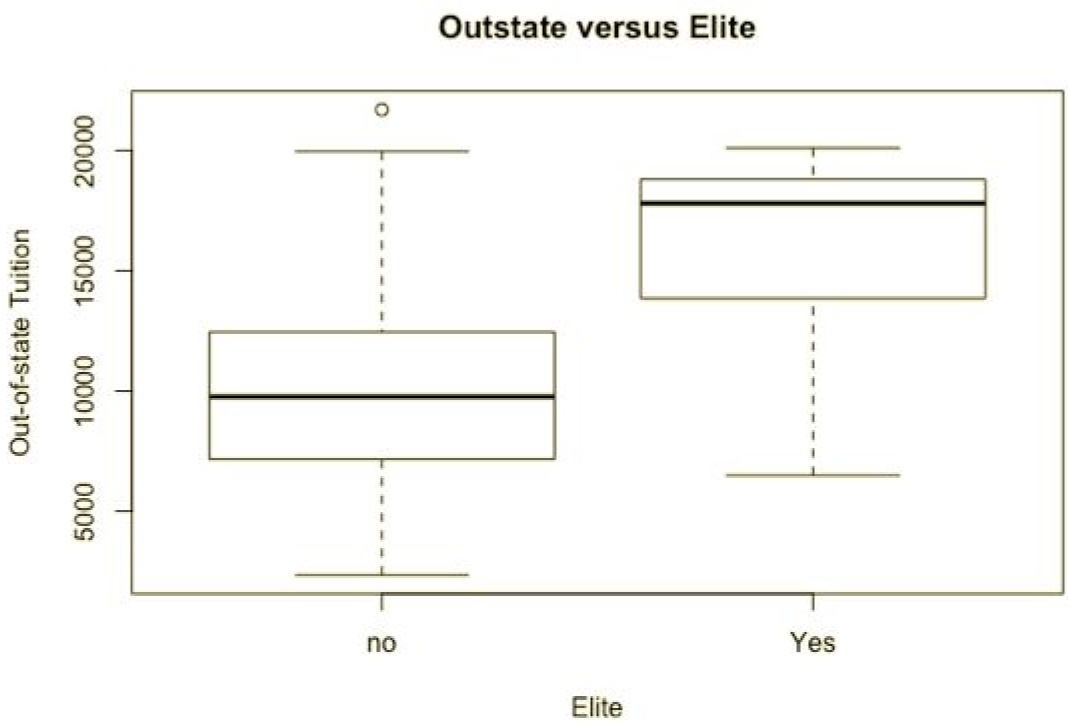


iv. Next we create a new qualitative variable, called Elite. By binning the Top10perc variable we divide universities into two groups based on whether or not the proportion of students coming from the top 10 % of their high school classes exceeds 50 %. As expected, the tuition costs for the elite colleges are substantially higher when comparing the median costs. Also note that the output in R produces **735 non-elite** and only **42 elite colleges**.

```
Elite = rep('no', nrow(college))
Elite[college$Top10perc > 60]='Yes'
Elite = as.factor(Elite)
college = data.frame(college, Elite)

# Use the summary() function to see how many elite universities
```

```
# there are. Now use the plot() function to produce
# side-by-side boxplots of Outstate versus Elite.
summary(college)

# Boxplot of Outstate by Elite
boxplot(Outstate~Elite,data=college, main="Outstate versus Elite",
      xlab="Elite", ylab="Out-of-state Tuition")
```

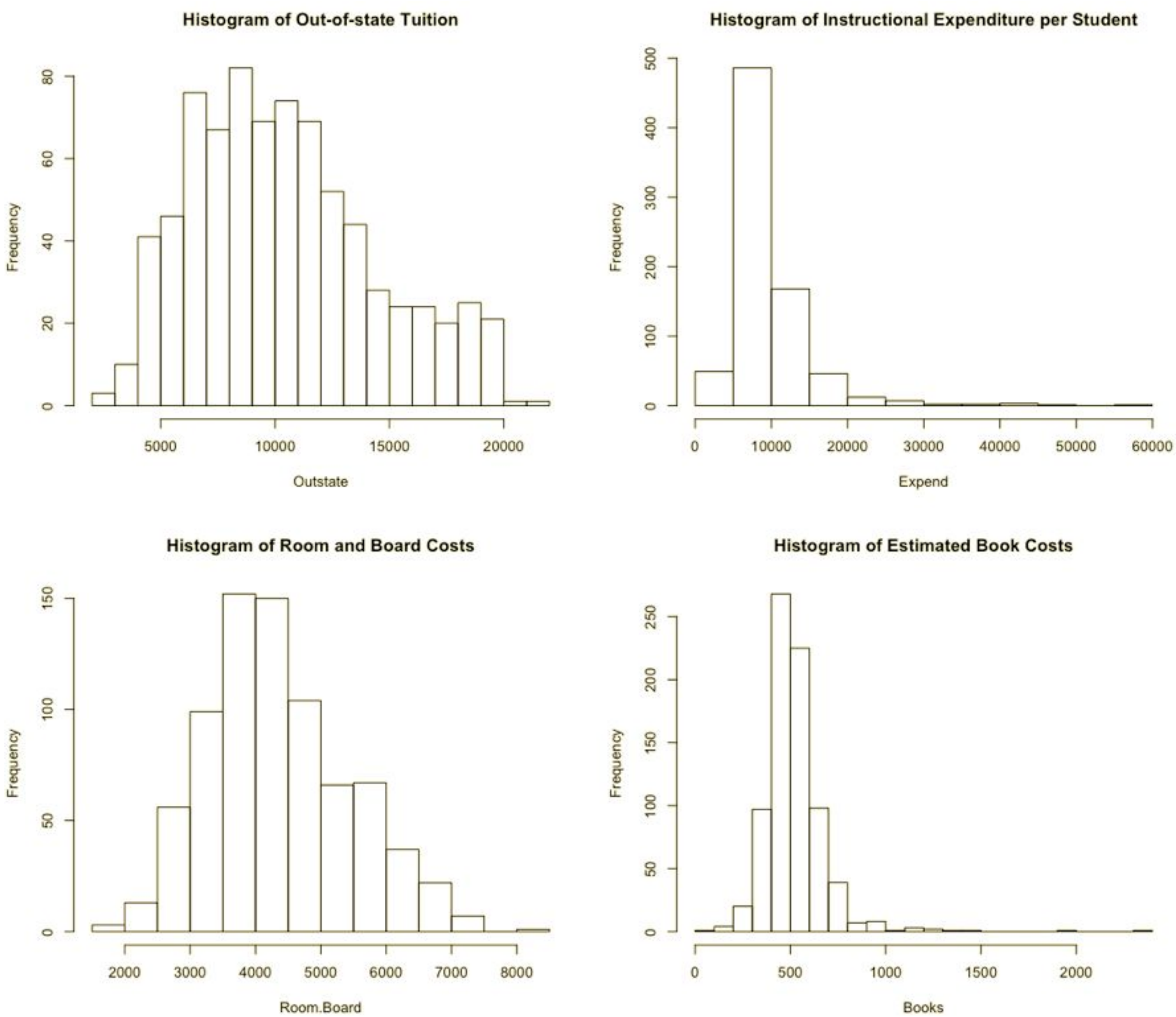
Fig 5



v. Using the hist() function we produce four histograms to evaluate college costs across the variables Outstate, Expend, Room.Board, and Books. The command **par(mfrow=c(2,2))** is used to break the display into four 2x2 segments, so that the plots can be viewed simultaneously. We alter the number of buckets for Outstate and Books variables to better represent the output. Book costs tend to be around $500 dollars per college at the median.

```
par(mfrow=c(2,2))
hist(Outstate, main="Histogram of Out-of-state Tuition", breaks=20)
hist(Expend, main="Histogram of Instructional Expenditure per Student")
hist(Room.Board, main="Histogram of Room and Board Costs")
hist(Books, main="Histogram of Estimated Book Costs", breaks=25)
```

Fig 6



vi. Next we continue to explore the data by ranking each school's acceptance rate. By evaluating the percentage of applicants who were accepted it is possible to approximate the school's overall popularity and rank. On this criteria alone we find that NU is ranked 30th out of the 777 schools.

```
# vi. Let's see where NU is ranked in terms of difficulty of being accepted.
# Calc the percentage accepted and add the calculated vector to the college data frame
pct.accept = Accept / Apps
college = data.frame(college, pct.accept)

# rank the accepted percentages
rank.difficulty.accepted = rank(pct.accept)
college = data.frame(college, rank.difficulty.accepted)

# output columns of interest, only for first n values
columns = c('Apps', 'Accept', 'pct.accept', 'rank.difficulty.accepted')
head(college[order(college$rank, decreasing= F),columns], n = 30)
```

Fig 7

```
> head(college[order(college$rank, decreasing= F),columns], n = 30)
                                        Apps Accept pct.accept rank.difficulty.accepted
Princeton University                   13218   2042  0.1544863                        1
Harvard University                     13865   2165  0.1561486                        2
Yale University                        10705   2453  0.2291453                        3
Amherst College                         4302    992  0.2305904                        4
Brown University                       12586   3239  0.2573494                        5
Georgetown University                  11115   2881  0.2591993                        6
Dartmouth College                       8587   2273  0.2647025                        7
Duke University                        13789   3893  0.2823265                        8
Columbia University                     6756   1930  0.2856720                        9
Williams College                        4186   1245  0.2974200                       10
Bowdoin College                         3356   1019  0.3036353                       11
Huron University                         600    197  0.3283333                       12
Washington and Lee University           3315   1096  0.3306184                       13
Spelman College                         3713   1237  0.3331538                       14
Massachusetts Institute of Technology   6411   2140  0.3338013                       15
University of Virginia                 15849   5384  0.3397060                       16
Talladega College                       4414   1500  0.3398278                       17
Rowan College of New Jersey             3820   1431  0.3746073                       18
Stockton College of New Jersey          4019   1579  0.3928838                       19
Davidson College                        2373    956  0.4028656                       20
Johns Hopkins University                8474   3446  0.4066557                       21
Montclair State University              5220   2128  0.4076628                       22
University of North Carolina at Chapel Hill 14596   5985  0.4100438                  23
Claremont McKenna College               1860    767  0.4123656                       24
Wesleyan University                     4772   1973  0.4134535                       25
University of California at Berkeley  19873   8252  0.4152368                       26
Harvey Mudd College                     1377    572  0.4153958                       27
University of Pennsylvania             12394   5232  0.4221397                       28
Wake Forest University                  5661   2392  0.4225402                       29
Northwestern University                12289   5200  0.4231426                       30
```

Exercise 9 of Section 2.4

Using the Auto data set we initially make sure that the missing values have been removed and the question marks are treated as missing element of the data matrix. Using header=T ensure that the initial row is treated as a header row. Note that there are only 5 rows containing missing observations. Running the head() function on the auto data set we can see that,

(a) the data table consists of the following **qualitative variables: cylinders, year, origin, name**, and the following **quantitative variables: mpg, displacement, horsepower, weight,** and **acceleration.**

```
auto = read.table('/Users/stevenfutter/Dropbox/NU/MACHINE_LEARNING/Auto.data.txt', header=T, na.strings='?')
```

```
dim(auto)    # returns > [1] 397   9
# only 5 rows contain missing obs
auto = na.omit(auto)
dim(auto)    # returns > [1] 392   9
head(auto)
```

(b) The range of each of the quantitative variables can be found by using the range() function, as below. Below we apply the range() function on each of the five quantitative variables.

```
range(mpg)
range(displacement)
range(horsepower)
range(weight)
range(acceleration)
```

Fig 8

```
> range(mpg)
[1]  9.0 46.6
> range(displacement)
[1]  68 455
> range(horsepower)
[1]  46 230
> range(weight)
[1] 1613 5140
> range(acceleration)
[1]  8.0 24.8
```

(c) Next we calculate the mean and standard deviation of the quantitative columns, only. Note that here we use sapply to apply the functions across each of the columns in the data.frame.

```
auto.quant = auto[c(1,3,4,5,6)]               # selects the quantitative columns, only
sapply(auto.quant, mean, na.rm = TRUE)  # sapply on auto.quant data.frame to calculate the mean
sapply(auto.quant, sd, na.rm = TRUE)      # sapply on auto.quant data.frame to calculate the SD
```

Fig 9

```
> auto.quant = auto[c(1,3,4,5,6)]        # selects the quantitative columns, only
> sapply(auto.quant, mean, na.rm = TRUE)  # sapply on auto.quant data.frame to calculate the mean
       mpg displacement   horsepower       weight acceleration
  23.44592    194.41199    104.46939   2977.58418     15.54133
> sapply(auto.quant, sd, na.rm = TRUE)    # sapply on auto.quant data.frame to calculate the standard deviation
       mpg displacement   horsepower       weight acceleration
  7.805007   104.644004    38.491160   849.402560     2.758864
```

(d) Next we remove the 10th through 85th observations and re-run mean, standard deviation, and range() functions on each quantitative variable. The negative value in -c(10,85) removes the rows from auto when added within the square brackets.

```
auto.quant.rm.10.85 = auto[-c(10,85),c(1,3,4,5,6)]
sapply(auto.quant.rm.10.85, range, na.rm = TRUE)   # sapply to calculate the range
sapply(auto.quant.rm.10.85, mean, na.rm = TRUE)   # sapply to calculate the mean
sapply(auto.quant.rm.10.85, sd, na.rm = TRUE)       # sapply to calculate the SD
```

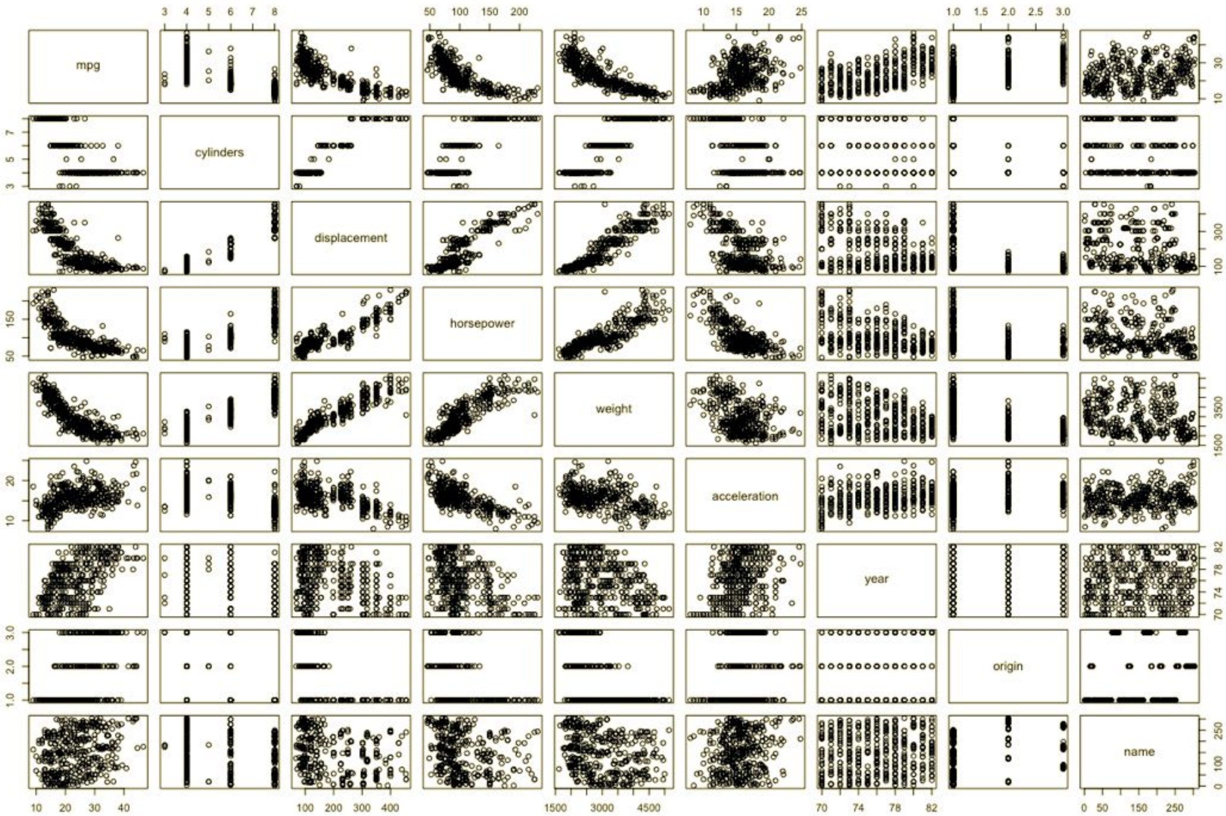
Fig 10

```
> auto.quant.rm.10.85 = auto[-c(10,85),c(1,3,4,5,6)]
> sapply(auto.quant.rm.10.85, range, na.rm = TRUE)  # sapply to calculate the range
      mpg displacement horsepower weight acceleration
[1,]  9.0           68         46   1613          8.0
[2,] 46.6          455        230   5140         24.8
> sapply(auto.quant.rm.10.85, mean, na.rm = TRUE)   # sapply to calculate the mean
       mpg displacement   horsepower       weight acceleration
  23.49436    193.51154    104.06923   2972.46923     15.56590
> sapply(auto.quant.rm.10.85, sd, na.rm = TRUE)     # sapply to calculate the standard deviation
       mpg displacement   horsepower       weight acceleration
  7.795198   104.140690    38.176331   848.512067     2.739672
```

(e) Here we use the full data set to investigate the predictors graphically. First we use the pairs() function to get a sense of the data frame, followed by some tabular data manipulation to find out which of the variables are most correlated with one another, then finally we create a 2x2 scatterplot matrix to present the findings.

```
# Use pairs to see correlations between all variables in auto data set. Fig 11 (below).
pairs(auto)
```

Fig 11



The pairs() output above shows that some variables appear to be highly correlated, but it is hard to decipher which are the most correlated variables from the scatterplots alone. Let's confirm what the most correlated variables are in the auto dataset.

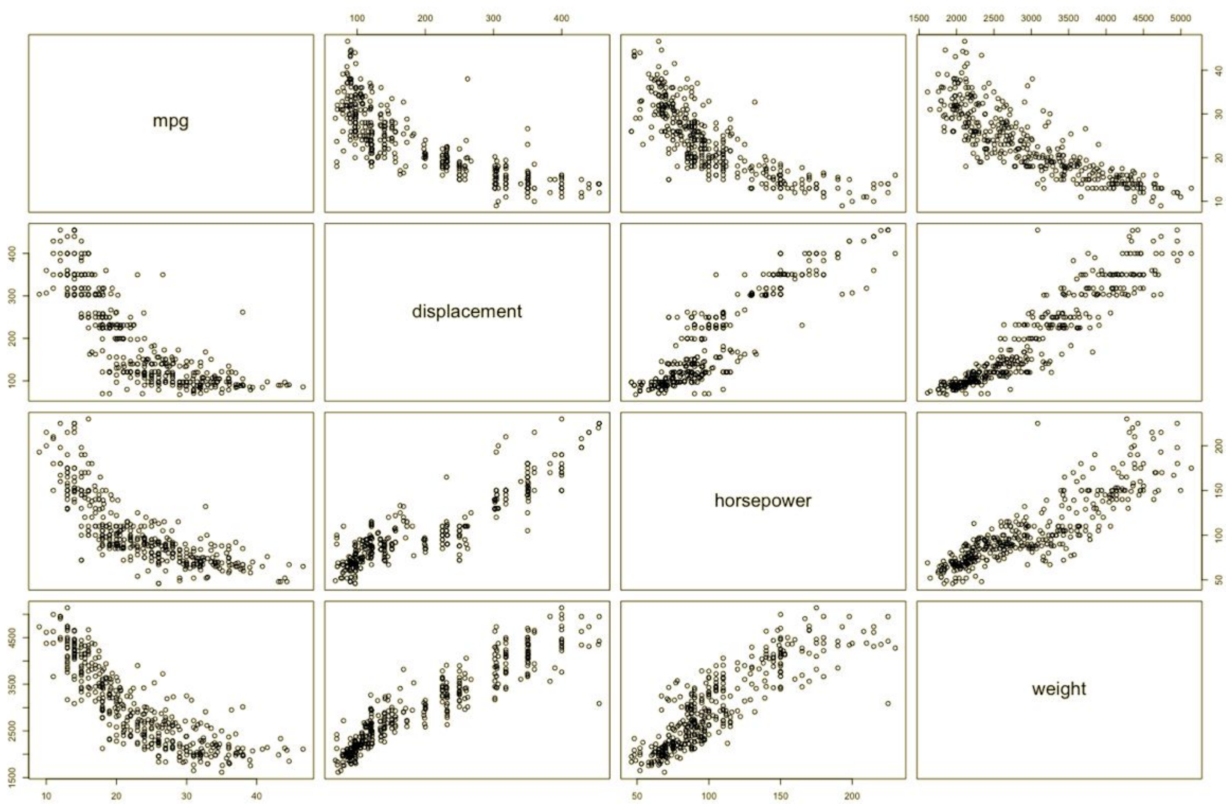
```
# To do this we use the dplyr and reshape2 libraries.
library(dplyr)
library(reshape2)
d.cor = as.matrix(cor(auto.quant))

# Note that using -abs(value) is necessary as correlation can be both -ve and +ve.
d.cor.melt = arrange(melt(d.cor), -abs(value))

# In the final step we manually select the rows of interest to remove the duplicate correlations.
d.cor.melt[c(6,8,10,12,14,16,18,20,22,24),]
```

Fig 12

```
> d.cor.melt[c(6,8,10,12,14,16,18,20,22,24),]
           Var1         Var2      value
6        weight displacement  0.9329944
8    horsepower displacement  0.8972570
10       weight   horsepower  0.8645377
12       weight          mpg -0.8322442
14 displacement          mpg -0.8051269
16   horsepower          mpg -0.7784268
18 acceleration   horsepower -0.6891955
20 acceleration displacement -0.5438005
22 acceleration          mpg  0.4233285
24 acceleration       weight -0.4168392
```

```
# Now that we have found the most correlated variables let's plot the scatterplot matrix
par(mfrow=c(2,2))
pairs(~ mpg + displacement + horsepower + weight, auto)
```

Fig 12



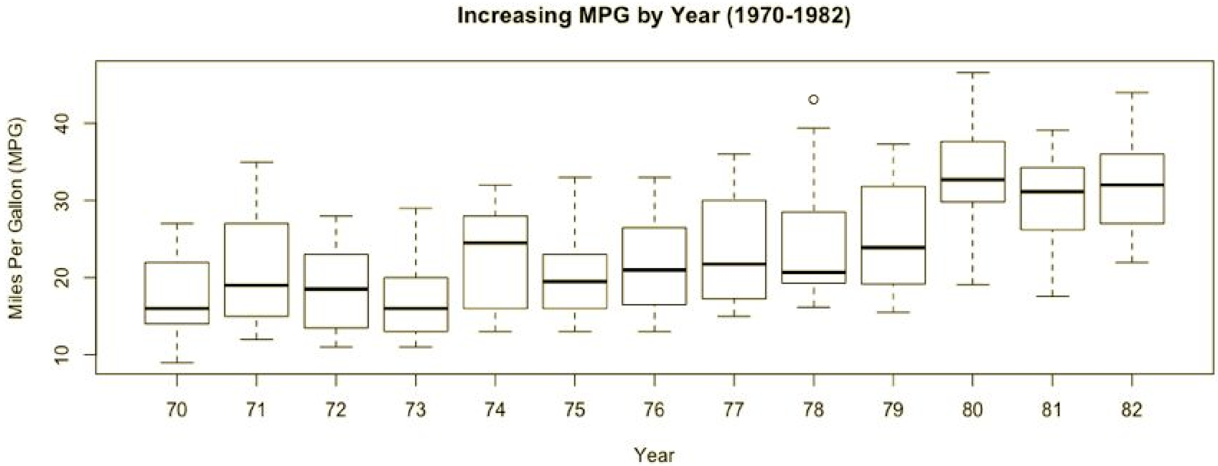
Interesting weight and displacement are the two most correlated variables in the auto dataset with a correlation of 0.933.

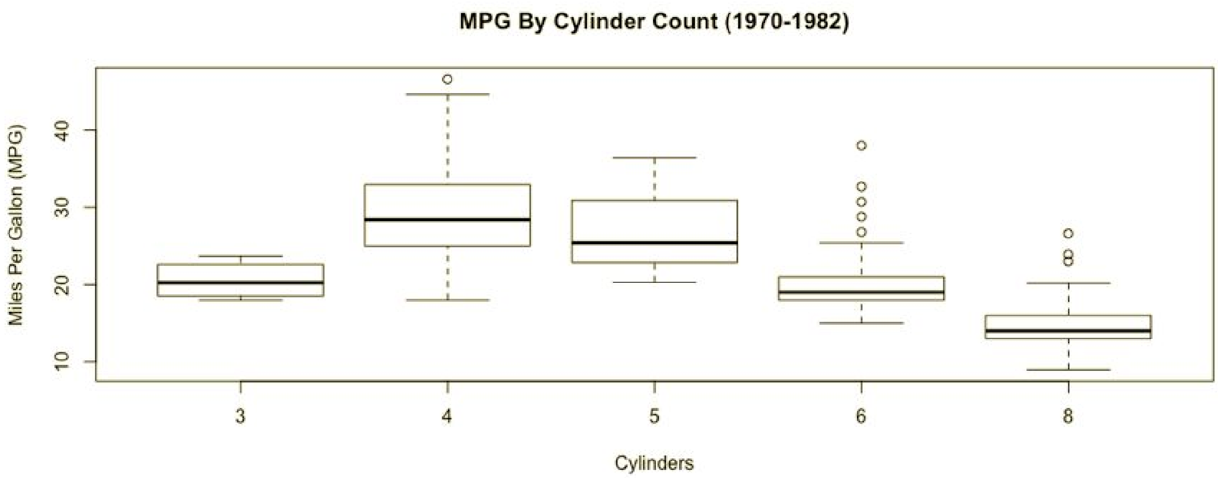
(f) There are other variables in the auto data set that may be useful when predicting gas mileage (mpg).

```
# create a 2x1 matrix boxplot display using both mpg on year and mpg against cylinder count
par(mfrow=c(2,1))
boxplot(mpg~year,data=auto, main="Increasing MPG by Year (1970-1982)", xlab="Year",
ylab="Miles Per Gallon (MPG)")
boxplot(mpg~cylinders,data=auto, main="MPG By Cylinder Count (1970-1982)",
xlab="Cylinders", ylab="Miles Per Gallon (MPG)")
```

As can be seen from the output below mpg has been slowly increasing during the years 1970 to 1982. Although there have been some variations in median mpg values, the general trend appears to be that of increasing mpg with each year. This makes sense as technology becomes more efficient over time and so cars are becoming more efficient, too. In addition, cylinder count is negatively correlated with gas mileage. With the exception of cars with three cylinders mpg is decreasing with increasing cylinder counts.

Fig 13


Increasing MPG by Year (1970-1982)
Miles Per Gallon (MPG)
10
20
30
40
70
71
72
73
74
75
76
77
78
79
80
81
82
Year


MPG By Cylinder Count (1970-1982)
Miles Per Gallon (MPG)
10
20
30
40
3
4
5
6
8
Cylinders