CODE

Carseats R Notebook

Cooper M. Stansbury (63919844)

1 Introduction

Here is my analysis of `data/Carseats_org.csv`.

1.1 Configuration

HIDE

```
library(leaps)
library(stringr)
library(caret)
library(ggplot2)
library(DataExplorer)
library(dplyr)
library(ggExtra)
library(RColorBrewer)
library(plotly)
library(corrplot)
library(htmltools)
library(MASS)
```

1.2 System Information

Due to the large number of libraries in use I have provided system information.

HIDE

```
sessionInfo()
```

```
R version 3.5.1 (2018-07-02)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: macOS High Sierra 10.13.6

Matrix products: default
BLAS: /System/Library/Frameworks/Accelerate.framework/Versions/A/Frameworks/vecLib.framework/Versions/A/libB
LAS.dylib
LAPACK: /anaconda3/lib/R/lib/libRblas.dylib

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] htmltools_0.3.6    corrplot_0.84      plotly_4.8.0       RColorBrewer_1.1-2 ggExtra_0.8
 [6] dplyr_0.8.0.1      DataExplorer_0.7.0 mgcv_1.8-26        nlme_3.1-137       caret_6.0-81
[11] ggplot2_3.1.0      lattice_0.20-38    leaps_3.0

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.0         lubridate_1.7.4    tidyr_0.8.2        class_7.3-15       assertthat_0.2.0
 [6] digest_0.6.18      ipred_0.9-8        foreach_1.4.4      mime_0.6           R6_2.4.0
[11] plyr_1.8.4         stats4_3.5.1       evaluate_0.12      httr_1.4.0         pillar_1.3.1
[16] rlang_0.3.1        lazyeval_0.2.1     rstudioapi_0.9.0   data.table_1.12.0  miniUI_0.1.1.1
[21] rpart_4.1-13       Matrix_1.2-15      rmarkdown_1.11     splines_3.5.1      gower_0.1.2
[26] stringr_1.4.0      htmlwidgets_1.3    igraph_1.2.4       munsell_0.5.0      shiny_1.2.0
[31] httpuv_1.4.5.1     compiler_3.5.1     xfun_0.4           pkgconfig_2.0.2    nnet_7.3-12
[36] tidyselect_0.2.5   tibble_2.0.1       gridExtra_2.3      prodlim_2018.04.18 codetools_0.2-16
[41] viridisLite_0.3.0  later_0.8.0        crayon_1.3.4       withr_2.1.2        MASS_7.3-51.1
[46] recipes_0.1.4      ModelMetrics_1.1.0 grid_3.5.1         xtable_1.8-3       jsonlite_1.6
[51] gtable_0.2.0       magrittr_1.5       scales_1.0.0       stringi_1.3.1      reshape2_1.4.3
[56] promises_1.0.1     timeDate_3043.102  generics_0.0.2     lava_1.6.5         iterators_1.0.10
[61] tools_3.5.1        glue_1.3.0         purrr_0.2.5        networkD3_0.4      parallel_3.5.1
[66] survival_2.43-3    yaml_2.2.0         colorspace_1.4-0   knitr_1.21
```

HIDE

```
sapply(c('repr', 'IRdisplay', 'IRkernel'), function(p) paste(packageVersion(p)))
```

```
    repr IRdisplay  IRkernel
 "0.19.2"   "0.7.0"  "0.8.15"
```

2 Data

First we inspect the raw records using `bash`. Always a good idea to look at things before you mash 'em into an IDE.

HIDE

```
head -n 5 data/Carseats_org.csv
```

```
"","Sales","CompPrice","Income","Advertising","Population","Price","ShelveLoc","Age","Education","Urban","US
"
"1",9.5,138,73,11,276,120,"Bad",42,17,"Yes","Yes"
"2",11.22,111,48,16,260,83,"Good",65,10,"Yes","Yes"
"3",10.06,113,35,10,269,80,"Medium",59,12,"Yes","Yes"
"4",7.4,117,100,4,466,97,"Medium",55,14,"Yes","Yes"
```

Now I load the data into `r`, and drop the "ID" column.

HIDE

```
carseats <- read.csv("data/Carseats_org.csv", header = T, stringsAsFactors = T)
drops <- c("X")
carseats <- carseats[ , !(names(carseats) %in% drops)]
head(carseats, 10)
```

Here I create two new data frames to manage numeric and categorical data.

HIDE

```
# get vectors of continuous and categorical cols
nums <- dplyr::select_if(carseats, is.numeric)
cats <- dplyr::select_if(carseats, is.factor)
nums[sample(nrow(nums), 10), ]
```

HIDE

```
cats[sample(nrow(cats), 10), ]
```

Let’s get some quick summaries of each:

HIDE

```
print('Numeric Summaries')
```

```
[1] "Numeric Summaries"
```

HIDE

```
summary(nums)
```

```
     Sales          CompPrice       Income        Advertising       Population        Price      
 Min.   : 0.000   Min.   : 77   Min.   : 21.00   Min.   : 0.000   Min.   : 10.0   Min.   : 24.0  
 1st Qu.: 5.390   1st Qu.:115   1st Qu.: 42.75   1st Qu.: 0.000   1st Qu.:139.0   1st Qu.:100.0  
 Median : 7.490   Median :125   Median : 69.00   Median : 5.000   Median :272.0   Median :117.0  
 Mean   : 7.496   Mean   :125   Mean   : 68.66   Mean   : 6.635   Mean   :264.8   Mean   :115.8  
 3rd Qu.: 9.320   3rd Qu.:135   3rd Qu.: 91.00   3rd Qu.:12.000   3rd Qu.:398.5   3rd Qu.:131.0  
 Max.   :16.270   Max.   :175   Max.   :120.00   Max.   :29.000   Max.   :509.0   Max.   :191.0  
      Age          Education   
 Min.   :25.00   Min.   :10.0  
 1st Qu.:39.75   1st Qu.:12.0  
 Median :54.50   Median :14.0  
 Mean   :53.32   Mean   :13.9  
 3rd Qu.:66.00   3rd Qu.:16.0  
 Max.   :80.00   Max.   :18.0  
```

HIDE

```
print('Categorical Summaries')
```

```
[1] "Categorical Summaries"
```

HIDE

```
summary(cats)
```

```
  ShelveLoc    Urban       US     
 Bad   : 96   No :118   No :142  
 Good  : 85   Yes:282   Yes:258  
 Medium:219                      
```

HIDE

```
str(nums)
```

```
'data.frame':	400 obs. of  8 variables:
 $ Sales      : num  9.5 11.22 10.06 7.4 4.15 ...
 $ CompPrice  : int  138 111 113 117 141 124 115 136 132 132 ...
 $ Income     : int  73 48 35 100 64 113 105 81 110 113 ...
 $ Advertising: int  11 16 10 4 3 13 0 15 0 0 ...
 $ Population : int  276 260 269 466 340 501 45 425 108 131 ...
 $ Price      : int  120 83 80 97 128 72 108 120 124 124 ...
 $ Age        : int  42 65 59 55 38 78 71 67 76 76 ...
 $ Education  : int  17 10 12 14 13 16 15 10 10 17 ...
```

HIDE

```
str(cats)
```

```
'data.frame':	400 obs. of  3 variables:
 $ ShelveLoc: Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 ...
 $ Urban    : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...
 $ US       : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...
```

2.1 Data Dimensionality

This command is to inspect the different data types in the data.

HIDE

```
str(carseats)
```

```
'data.frame':	400 obs. of  11 variables:
 $ Sales      : num  9.5 11.22 10.06 7.4 4.15 ...
 $ CompPrice  : int  138 111 113 117 141 124 115 136 132 132 ...
 $ Income     : int  73 48 35 100 64 113 105 81 110 113 ...
 $ Advertising: int  11 16 10 4 3 13 0 15 0 0 ...
 $ Population : int  276 260 269 466 340 501 45 425 108 131 ...
 $ Price      : int  120 83 80 97 128 72 108 120 124 124 ...
 $ ShelveLoc  : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 ...
 $ Age        : int  42 65 59 55 38 78 71 67 76 76 ...
 $ Education  : int  17 10 12 14 13 16 15 10 10 17 ...
 $ Urban      : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...
 $ US         : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...
```

HIDE

```
print("")
```

```
[1] ""
```

HIDE

```
print(paste('Number of Columns:', ncol(carseats)))
```

```
[1] "Number of Columns: 11"
```

HIDE

```
print(paste('Number of Numeric Columns:', ncol(nums)))
```

```
[1] "Number of Numeric Columns: 8"
```

HIDE

```
print(paste('Number of Categorical Columns:', ncol(cats)))
```

```
[1] "Number of Categorical Columns: 3"
```

HIDE

```r
dim(carseats)
```

```
[1] 400  11
```

HIDE

```r
dim(nums)
```

```
[1] 400   8
```

HIDE

```r
dim(cats)
```

```
[1] 400   3
```

Here's a quick way to examine general properties of the data:

HIDE

```r
DataExplorer::introduce(data=carseats)
```

Finally, I want to look at the first and last rows of the data set. Just to be safe:

HIDE

```r
head(carseats, 2)
```

HIDE

```r
tail(carseats, 2)
```

HIDE

```r
head(nums, 2)
```

HIDE

```r
tail(nums, 2)
```

HIDE

```r
head(cats, 2)
```

HIDE

```r
tail(cats, 2)
```

3 Numeric Plotting

I start out with a few general scatter plots.

HIDE

```
plot_ly(data=carseats,
        x=~Age,
        y=~Sales,
        mode = 'markers',
        type = 'scatter',
        color=~ShelveLoc) %>%
            layout(title = "Age, Shelf Location, and Sales Scatter Plot", width=900)
```

```
Specifying width/height in layout() is now deprecated.
Please specify in ggplotly() or plot_ly()
```

This plot below shows good separation and a weak linear trend. These variables are worth investigating.

HIDE

```
plot_ly(data=carseats,
        x=~Price,
        y=~Sales,
        mode = 'markers',
        type = 'scatter',
        color=~ShelveLoc) %>%
            layout(title = "Price, Shelf Location, and Sales Scatter Plot", width=900)
```

```
Specifying width/height in layout() is now deprecated.
Please specify in ggplotly() or plot_ly()
```

Here we inspect the density of the ‘Price vs Sales’ relationship:

HIDE

```
plot_ly(data=carseats,
        x=~Price,
        y=~Sales,
        mode = 'markers',
        size = ~Price,
        type = 'scatter',
        colors = "Dark2",
        alpha = .6) %>%
            layout(title = "Price, US, and Sales Scatter Plot", width=900)
```

```
Specifying width/height in layout() is now deprecated.
Please specify in ggplotly() or plot_ly()`line.width` does not currently support multiple values.`line.width
` does not currently support multiple values.
```

4 Normalization

I choose to normalize the numeric data in order to be able to plot each variable on the same scale. This will allow me to investigate the variation of each predictor relative to Sales.

HIDE

```
preObj <- preProcess(nums, method=c("center", "scale"))
scaled.nums <- predict(preObj, nums)
head(scaled.nums, 2)
```

HIDE

```
tail(scaled.nums, 2)
```

HIDE

```
str(scaled.nums)
```

```
'data.frame':   400 obs. of  8 variables:
 $ Sales      : num  0.7095 1.3185 0.9078 -0.0341 -1.1849 ...
 $ CompPrice  : num  0.849 -0.911 -0.781 -0.52 1.045 ...
 $ Income     : num  0.155 -0.738 -1.203 1.12 -0.166 ...
 $ Advertising: num  0.656 1.408 0.506 -0.396 -0.547 ...
 $ Population : num  0.0757 -0.0328 0.0282 1.3649 0.51 ...
 $ Price      : num  0.178 -1.385 -1.512 -0.794 0.515 ...
 $ Age        : num  -0.699 0.721 0.35 0.104 -0.946 ...
 $ Education  : num  1.183 -1.4882 -0.725 0.0382 -0.3434 ...
```

HIDE

```
print("")
```

```
[1] ""
```

HIDE

```
summary(scaled.nums)
```

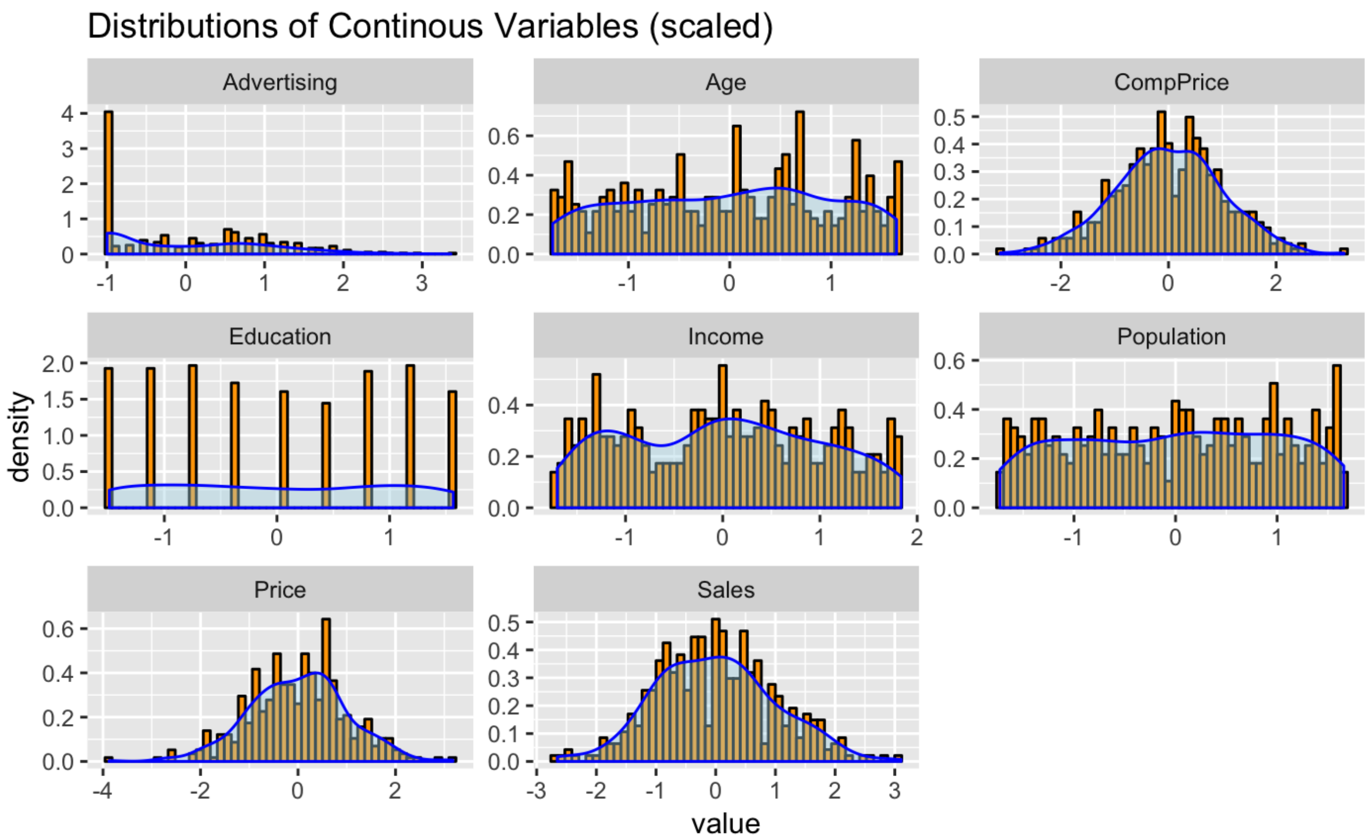
```
     Sales              CompPrice              Income            Advertising          Population
 Min.   :-2.65440   Min.   :-3.12856   Min.   :-1.70290   Min.   :-0.9977   Min.   :-1.72918
 1st Qu.:-0.74584   1st Qu.:-0.65049   1st Qu.:-0.92573   1st Qu.:-0.9977   1st Qu.:-0.85387
 Median :-0.00224   Median : 0.00163   Median : 0.01224   Median :-0.2459   Median : 0.04858
 Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.00000
 3rd Qu.: 0.64575   3rd Qu.: 0.65375   3rd Qu.: 0.79834   3rd Qu.: 0.8067   3rd Qu.: 0.90693
 Max.   : 3.10670   Max.   : 3.26225   Max.   : 1.83458   Max.   : 3.3630   Max.   : 1.65671
     Price               Age               Education
 Min.   :-3.87702   Min.   :-1.74827   Min.   :-1.48825
 1st Qu.:-0.66711   1st Qu.:-0.83779   1st Qu.:-0.72504
 Median : 0.05089   Median : 0.07268   Median : 0.03816
 Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.00000
 3rd Qu.: 0.64219   3rd Qu.: 0.78255   3rd Qu.: 0.80137
 Max.   : 3.17633   Max.   : 1.64673   Max.   : 1.56457
```

4.1 Distributions

Here are scaled distributions (histograms and density plots) for each numeric variable, including Sales. The variables relating to money ($) tend to be approximately normal. Many other variables tend to be approximately uniform, which does not bode well for their predictive power.

HIDE

```
scaled.nums %>%
    tidyr::gather() %>%
        ggplot(aes(x=value,y=..density..))+
            ggtitle('Distributions of Continous Variables (scaled)') +
            facet_wrap(~ key, scales = "free") +
            geom_histogram(fill=I("orange"), col=I("black"), bins = 50) +
            facet_wrap(~ key, scales = "free") +
            geom_density(color="blue", fill='light blue', alpha = 0.4)
```



Here we plot all numeric variables against their distributions. This is just another way to examine the information shown above.

HIDE

```
scaled.nums %>%
    tidyr::gather() %>%
        plot_ly(x=~key, y=~value,
                type = "box",
                boxpoints = "all",
                jitter = 0.4,
                pointpos = 0,
                color = ~key,
                colors = "Dark2") %>%
                      subplot(shareX = TRUE)  %>%
                            layout(title = "Numeric Variable Distributions (scaled)",
                                  yaxis=list(title='Standard Deviation'),
                                  xaxis=list(title='Variable'),
                                  autosize=FALSE,
                                  width=900,
                                  height=500)
```

```
Specifying width/height in layout() is now deprecated.
Please specify in ggplotly() or plot_ly()
```

4.2 Scatterplots

Here we plot all numeric variables against Sales (scaled). This allows us to investigate possible linear relationships between that variable and Sales. As shown below, only 'Price' appears to have a linear relationship worth investigating. This took me so long to figure out.

HIDE

```
numeric.scatterplots <- htmltools::tagList()
count = 1
for (i in names(scaled.nums[,-1])) {

  numeric.scatterplots[[count]] <- plot_ly(scaled.nums, x=scaled.nums[,i], y=scaled.nums$Sales,
                    colors = 'RdYlGn',
                    mode = 'markers',
                    type = 'scatter',
                    size = scaled.nums$Sales^2,
                    color = scaled.nums$Sales,
                    marker = list(line = list(color = 'black',width = 2)),
                    name=paste(i)) %>%
          layout(title = paste(i, "vs Sales (scaled)<br>Size=Sales^2"),
                              yaxis=list(title='Sales'),
                              xaxis=list(title=i),
                              showlegend = FALSE)

  count = count + 1

}
numeric.scatterplots
```

```
`line.width` does not currently support multiple values.`line.width` does not currently support multiple val
ues.`line.width` does not currently support multiple values.`line.width` does not currently support multiple
values.`line.width` does not currently support multiple values.`line.width` does not currently support multi
ple values.`line.width` does not currently support multiple values.`line.width` does not currently support m
ultiple values.`line.width` does not currently support multiple values.`line.width` does not currently suppo
rt multiple values.`line.width` does not currently support multiple values.`line.width` does not currently s
upport multiple values.`line.width` does not currently support multiple values.`line.width` does not current
ly support multiple values.
```

By adding naive regression lines to a few scatter plots we can confirm our suspicions:

HIDE

```
fit.Pop <- lm(Sales ~ Population, data = scaled.nums)
fit.Age <- lm(Sales ~ Age, data = scaled.nums)
fit.CompPrice <- lm(Sales ~ CompPrice, data = scaled.nums)
fit.Price <- lm(Sales ~ Price, data = scaled.nums)
regression.scatterplots <- htmltools::tagList()
regression.scatterplots[[1]] <-  plot_ly(scaled.nums,
          x = ~Population,
          name = 'Population vs Sales Regression Line') %>%
              add_markers(y = ~Sales,
                    name = 'Population vs Sales Observations') %>%
                            add_lines(x = ~Population,
                                  y = fitted(fit.Pop)) %>%
                                        layout(title = "Population vs Sales",
                                               yaxis=list(title='Sales',
                                               xaxis=list(title='Population')),
                                               showlegend = FALSE)


regression.scatterplots[[2]] <-  plot_ly(scaled.nums,
          x = ~Age,
          name = 'Age vs Sales Regression Line') %>%
              add_markers(y = ~Sales,
                    name = 'Age vs Sales Observations') %>%
                            add_lines(x = ~Age,
                                  y = fitted(fit.Age)) %>%
                                        layout(title = "Age vs Sales",
                                               yaxis=list(title='Sales',
                                               xaxis=list(title='Age')),
                                               showlegend = FALSE)
regression.scatterplots[[3]] <-  plot_ly(scaled.nums,
          x = ~CompPrice,
          name = 'CompPrice vs Sales Regression Line') %>%
              add_markers(y = ~Sales,
                    name = 'CompPrice vs Sales Observations') %>%
                            add_lines(x = ~CompPrice,
                                  y = fitted(fit.CompPrice)) %>%
                                        layout(title = "CompPrice vs Sales",
                                               yaxis=list(title='Sales',
                                               xaxis=list(title='CompPrice')),
                                               showlegend = FALSE)
regression.scatterplots[[4]] <-  plot_ly(scaled.nums,
          x = ~Price,
          name = 'Price vs Sales Regression Line') %>%
              add_markers(y = ~Sales,
                    name = 'Price vs Sales Observations') %>%
                            add_lines(x = ~Price,
                                  y = fitted(fit.Price)) %>%
                                        layout(title = "Price vs Sales",
                                               yaxis=list(title='Sales',
                                               xaxis=list(title='Price')),
                                               showlegend = FALSE)
regression.scatterplots
```

Let's compare the slopes:

HIDE

```
scaled.nums %>%
    plot_ly(y = ~Sales) %>%
      add_lines(x= ~Population, y = fitted(fit.Pop),
                name = "fit.Pop slope", line = list(shape = "linear")) %>%
      add_lines(x= ~Age, y = fitted(fit.Age),
                name = "fit.Age slope", line = list(shape = "linear")) %>%
      add_lines(x= ~CompPrice, y = fitted(fit.CompPrice),
                name = "fit.CompPrice slope", line = list(shape = "linear")) %>%
      add_lines(x= ~Price, y = fitted(fit.Price),
                name = "fit.Price slope", line = list(shape = "linear")) %>%

    layout(title = "Regression Lines vs Sales (scaled)",
           autosize=FALSE,
           width=900,
           yaxis=list(title='Sales'),
           xaxis=list(title='Scaled Numeric Variable'))
```

```
Specifying width/height in layout() is now deprecated.
Please specify in ggplotly() or plot_ly()
```

Here's a pretty graphic that doesn't help me understand anything about the data.

HIDE

```
y = scaled.nums$Sales
x = scaled.nums$Price
s <- subplot(
  plot_ly(x = x, color = I("black"), type = 'histogram'),
  plotly_empty(),
  plot_ly(x = x, y = y, type = 'histogram2dcontour', showscale = F),
  plot_ly(y = y, color = I("black"), type = 'histogram'),
  nrows = 2, heights = c(0.2, 0.8), widths = c(0.8, 0.2),
  shareX = TRUE, shareY = TRUE, titleX = FALSE, titleY = FALSE)
```

```
No trace type specified and no positional attributes specifiedNo trace type specified:
  Based on info supplied, a 'scatter' trace seems appropriate.
  Read more about this trace type -> https://plot.ly/r/reference/#scatter
No scatter mode specifed:
  Setting the mode to markers
  Read more about this attribute -> https://plot.ly/r/reference/#scatter-mode
```

HIDE

```
layout(s, showlegend = FALSE, autosize=FALSE,
           width=700,
           height=500,
           yaxis=list(title='Sales'),
           xaxis=list(title='Price'))
```

```
Specifying width/height in layout() is now deprecated.
Please specify in ggplotly() or plot_ly()
```

5 Categorical Plotting

First, let's create a data frame that we can use:

HIDE

```
categorical.by.sales = cbind(Sales = scaled.nums$Sales, cats)
str(categorical.by.sales)
```

```
'data.frame':   400 obs. of  4 variables:
 $ Sales    : num  0.7095 1.3185 0.9078 -0.0341 -1.1849 ...
 $ ShelveLoc: Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 ...
 $ Urban    : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...
 $ US       : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...
```

Here we can see all categorical by Sales. We suspected that 'ShelveLoc' would be important based on one of the early scatter plots. It seems that this is the case.

HIDE

```
categorical.boxplots <- htmltools::tagList()
count = 1
for (i in names(categorical.by.sales[,-1])) {

  categorical.boxplots[[count]] <- plot_ly(categorical.by.sales, x=categorical.by.sales[,i], y=categorical.by.

                        type = "box",
                        boxpoints = "all",
                        jitter = .2,
                        pointpos = 0,
                        color =categorical.by.sales[,i],
                        colors='Set1',
                        name=paste(i)) %>%
                            layout(title = paste(i, "vs Sales (scaled)"),
                                  showlegend = TRUE,
                                  yaxis=list(title='Sales Standard Deviation'),
                                  xaxis=list(title=i))

  count = count + 1

}
categorical.boxplots
```

Here's the same thing, but more musically:

HIDE

```
categorical.violins <- htmltools::tagList()
count = 1
for (i in names(categorical.by.sales[,-1])) {

  categorical.violins[[count]] <- plot_ly(categorical.by.sales, x=categorical.by.sales[,i], y=categorical.by.s

                      split = categorical.by.sales[,i],
                      type = 'violin',
                      colors='Set1',
                      name=paste(i),
                      box = list(visible = TRUE),
                      meanline = list(visible = TRUE)) %>%
                          layout(xaxis = list(title = "US"),
                              yaxis = list(title = "Sales",zeroline = FALSE))

  count = count + 1

}
categorical.violins
```

6 Linear Regression

First, let's merge the data set into a single data frame

HIDE

```
scaled.merged <- cbind(categorical.by.sales[,-1], scaled.nums)
str(scaled.merged)
```

```
'data.frame':   400 obs. of  11 variables:
 $ ShelveLoc  : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 ...
 $ Urban      : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...
 $ US         : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...
 $ Sales      : num  0.7095 1.3185 0.9078 -0.0341 -1.1849 ...
 $ CompPrice  : num  0.849 -0.911 -0.781 -0.52 1.045 ...
 $ Income     : num  0.155 -0.738 -1.203 1.12 -0.166 ...
 $ Advertising: num  0.656 1.408 0.506 -0.396 -0.547 ...
 $ Population : num  0.0757 -0.0328 0.0282 1.3649 0.51 ...
 $ Price      : num  0.178 -1.385 -1.512 -0.794 0.515 ...
 $ Age        : num  -0.699 0.721 0.35 0.104 -0.946 ...
 $ Education  : num  1.183 -1.4882 -0.725 0.0382 -0.3434 ...
```

HIDE

```
head(nums, 2)
```

HIDE

```
tail(nums, 2)
```

HIDE

```
head(scaled.merged, 2)
```

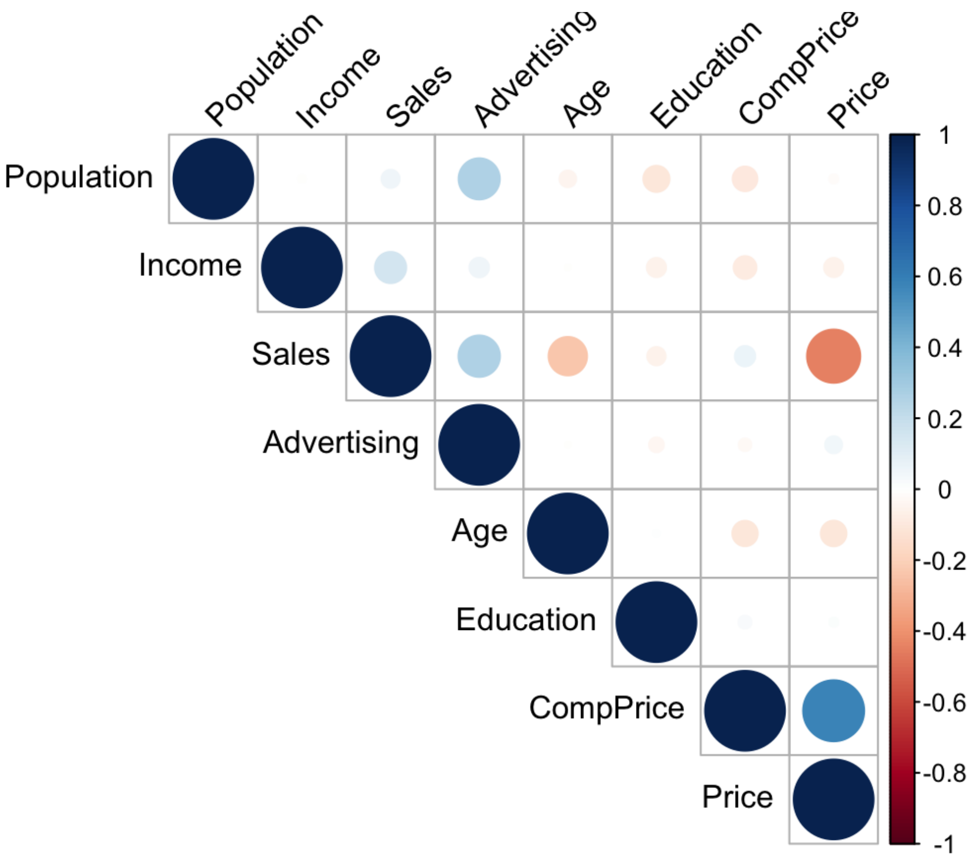
HIDE

```
tail(scaled.merged, 2)
```

First, let's look at some things that may give us trouble. Luckily it looks like the only serious correlation is with our dependent variable. We'll want to watch the 'Price' vs 'CompPrice' relationship.

HIDE

```
res <- cor(scaled.nums)
corrplot(res, type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45)
```



It appears that residuals are roughly symmetrical around 0. That's strange. Mostly due to a relatively poor overall fit. Note how close to zero most of the coefficient estimates are.

HIDE

```
simple.lm <- lm(Sales~., data=scaled.merged)
simple.summary <- summary(simple.lm)
print(simple.summary)
```

```

Call:
lm(formula = Sales ~ ., data = scaled.merged)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.01598 -0.24463  0.00748  0.23496  1.20797 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)     -0.73292    0.05999 -12.217  < 2e-16 ***
ShelveLocGood    1.71742    0.05422  31.678  < 2e-16 ***
ShelveLocMedium  0.69286    0.04465  15.516  < 2e-16 ***
UrbanYes         0.04351    0.04000   1.088    0.277    
USYes           -0.06519    0.05306  -1.229    0.220    
CompPrice        0.50397    0.02252  22.378  < 2e-16 ***
Income           0.15660    0.01828   8.565 2.58e-16 ***
Advertising      0.28987    0.02619  11.066  < 2e-16 ***
Population       0.01085    0.01933   0.561    0.575    
Price           -0.79946    0.02239 -35.700  < 2e-16 ***
Age             -0.26413    0.01825 -14.472  < 2e-16 ***
Education       -0.01958    0.01830  -1.070    0.285    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3608 on 388 degrees of freedom
Multiple R-squared:  0.8734,	Adjusted R-squared:  0.8698 
F-statistic: 243.4 on 11 and 388 DF,  p-value: < 2.2e-16
```

6.1 Linear Models and Subsets

Let's do the same thing, but control the subsets using `leaps`

HIDE

```
regfit.full=regsubsets(Sales~., data=scaled.merged, nvmax=5)
reg.summary=summary(regfit.full)
print(reg.summary)
```

```
Subset selection object
Call: regsubsets.formula(Sales ~ ., data = scaled.merged, nvmax = 5)
11 Variables  (and intercept)
                Forced in Forced out
ShelveLocGood       FALSE      FALSE
ShelveLocMedium     FALSE      FALSE
UrbanYes            FALSE      FALSE
USYes               FALSE      FALSE
CompPrice           FALSE      FALSE
Income              FALSE      FALSE
Advertising         FALSE      FALSE
Population          FALSE      FALSE
Price               FALSE      FALSE
Age                 FALSE      FALSE
Education           FALSE      FALSE
1 subsets of each size up to 5
Selection Algorithm: exhaustive
         ShelveLocGood ShelveLocMedium UrbanYes USYes CompPrice Income Advertising Population Price Age
1  ( 1 ) "*"           " "             " "      " "   " "       " "    " "         " "        " "   " "
2  ( 1 ) "*"           " "             " "      " "   " "       " "    " "         " "        "*"   " "
3  ( 1 ) "*"           " "             " "      " "   "*"       " "    " "         " "        "*"   " "
4  ( 1 ) "*"           " "             " "      " "   "*"       " "    "*"         " "        "*"   " "
5  ( 1 ) "*"           "*"             " "      " "   "*"       " "    "*"         " "        "*"   " "
         Education
1  ( 1 ) " "      
2  ( 1 ) " "      
3  ( 1 ) " "      
4  ( 1 ) " "      
5  ( 1 ) " "      
```

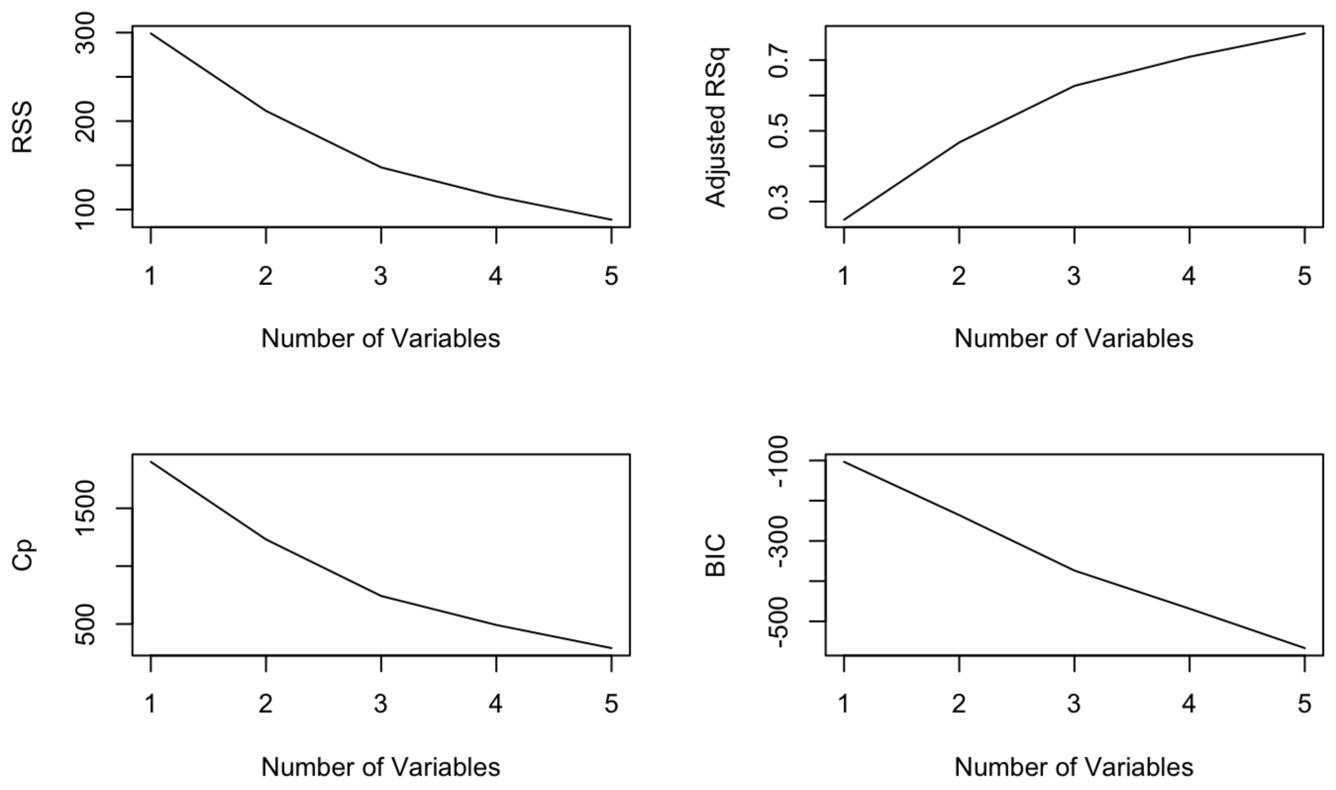
We'll just take code straight from the example on Canvas…

HIDE

```
par(mfrow=c(2,2))
plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",type="l")
plot(reg.summary$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="l")
```

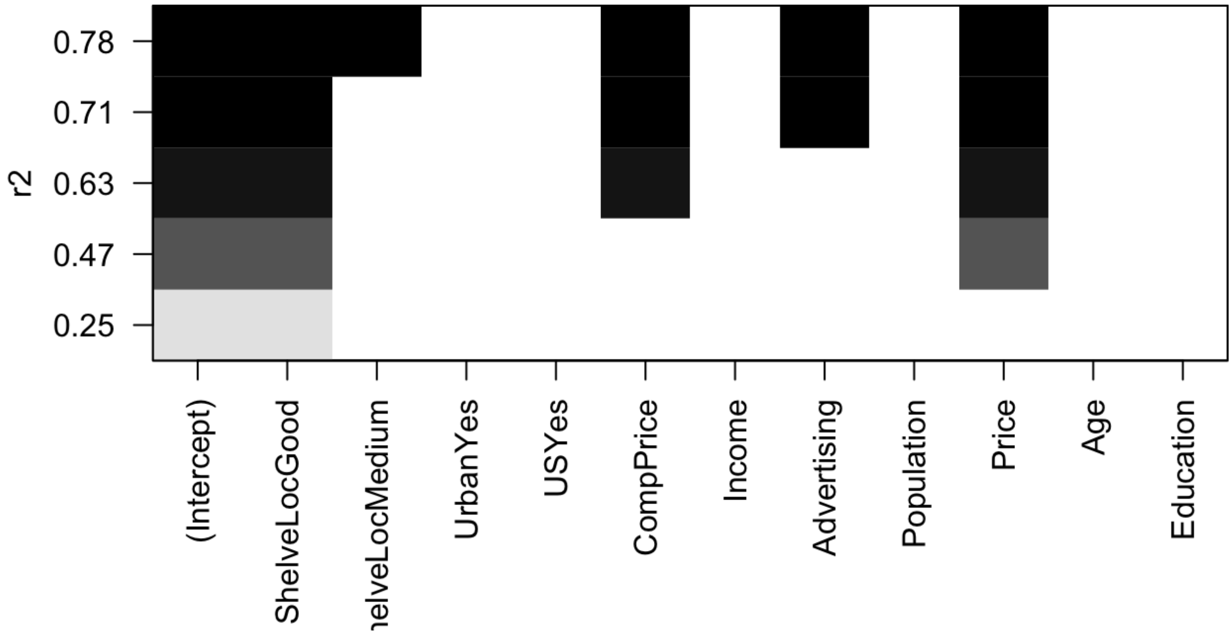
HIDE

```
plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp",type='l')
plot(reg.summary$bic,xlab="Number of Variables",ylab="BIC",type='l')
```



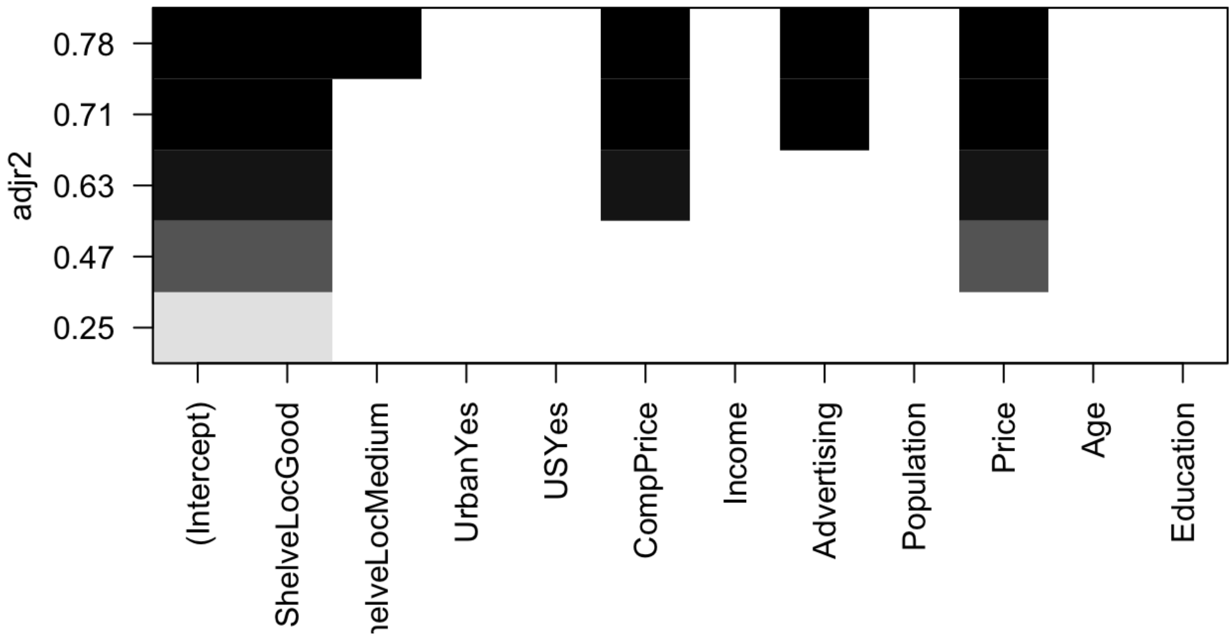
HIDE

```
plot(regfit.full,scale="r2")
```



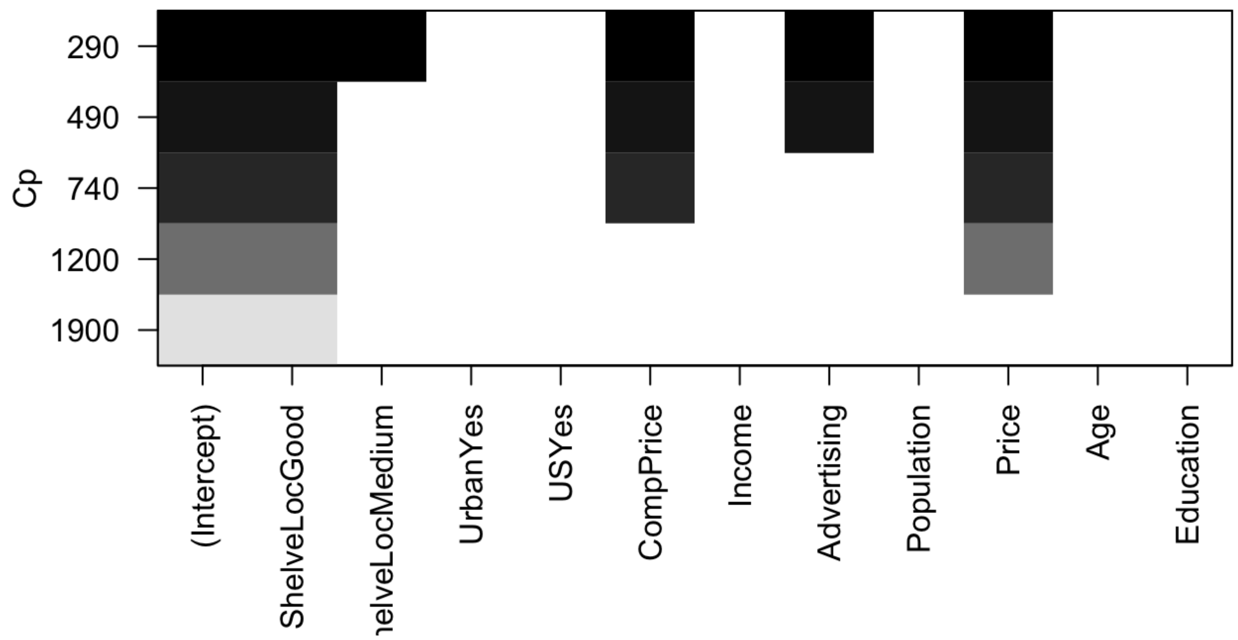
HIDE

```
plot(regfit.full,scale="adjr2")
```



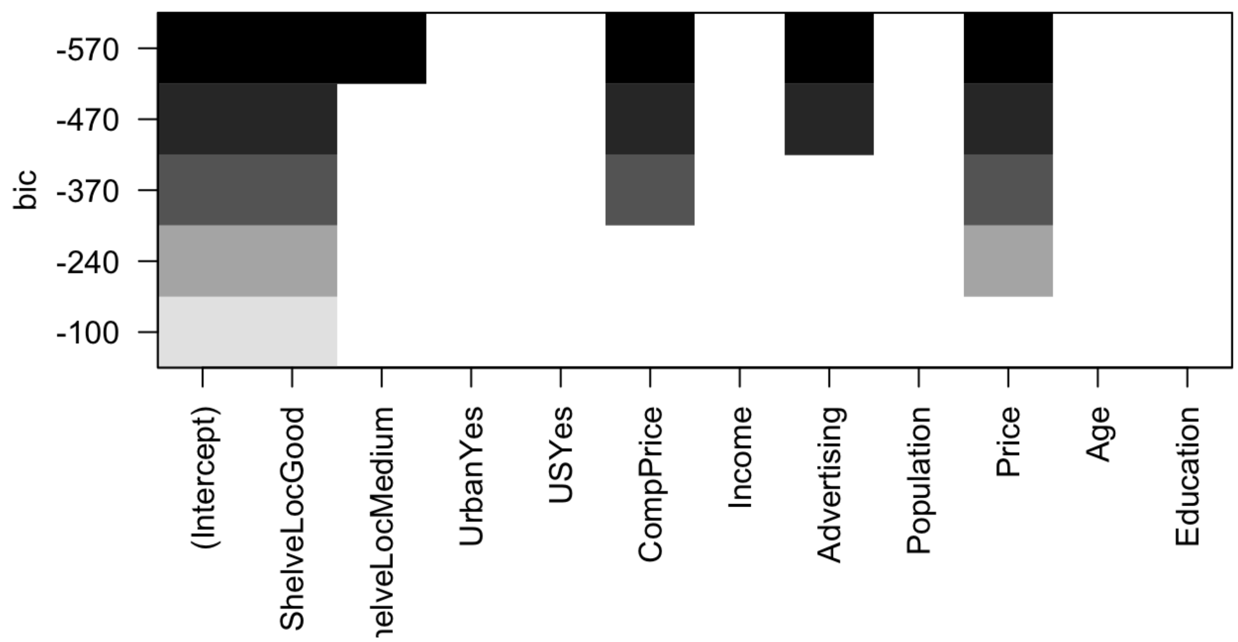
HIDE

```
plot(regfit.full,scale="Cp")
```



HIDE

```
plot(regfit.full,scale="bic")
```



7 Interaction Terms

Here we define a new model with some interaction terms: a. Income and Advertising b. Income and CompPrice c. Price and Age

HIDE

```
interaction.lm <- lm(Sales~. + Income*Advertising + Income*CompPrice + Price*Age, data=scaled.merged)
interaction.summary <- summary(interaction.lm)
print(interaction.summary)
```

```

Call:
lm(formula = Sales ~ . + Income * Advertising + Income * CompPrice + 
    Price * Age, data = scaled.merged)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.04967 -0.23955 -0.00936  0.24591  1.19774 

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)    
(Intercept)        -0.730051   0.059236 -12.324  < 2e-16 ***
ShelveLocGood       1.714324   0.053519  32.032  < 2e-16 ***
ShelveLocMedium     0.680563   0.044177  15.405  < 2e-16 ***
UrbanYes            0.045257   0.039380   1.149  0.25117    
USYes              -0.069622   0.052329  -1.330  0.18415    
CompPrice           0.507567   0.022131  22.935  < 2e-16 ***
Income              0.152164   0.018025   8.442 6.43e-16 ***
Advertising         0.289455   0.025756  11.238  < 2e-16 ***
Population          0.007224   0.018985   0.381  0.70377    
Price              -0.797242   0.022003 -36.233  < 2e-16 ***
Age                -0.260783   0.017978 -14.505  < 2e-16 ***
Education          -0.024248   0.018063  -1.342  0.18027    
Income:Advertising  0.046378   0.018170   2.552  0.01108 *  
CompPrice:Income   -0.059184   0.018911  -3.130  0.00188 ** 
Price:Age           0.013171   0.017912   0.735  0.46259    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3538 on 385 degrees of freedom
Multiple R-squared:  0.8792,	Adjusted R-squared:  0.8748 
F-statistic: 200.1 on 14 and 385 DF,  p-value: < 2.2e-16
```

HIDE

```
interaction.lm.subsets <- regsubsets(Sales~. + Income*Advertising + Income*CompPrice + Price*Age,
                                     data=scaled.merged, nvmax=5)
interaction.subsets.summary <- summary(interaction.lm.subsets)
print(interaction.subsets.summary)
```

```
Subset selection object
Call: regsubsets.formula(Sales ~ . + Income * Advertising + Income *
    CompPrice + Price * Age, data = scaled.merged, nvmax = 5)
14 Variables  (and intercept)
                   Forced in Forced out
ShelveLocGood          FALSE      FALSE
ShelveLocMedium        FALSE      FALSE
UrbanYes               FALSE      FALSE
USYes                  FALSE      FALSE
CompPrice              FALSE      FALSE
Income                 FALSE      FALSE
Advertising            FALSE      FALSE
Population             FALSE      FALSE
Price                  FALSE      FALSE
Age                    FALSE      FALSE
Education              FALSE      FALSE
Income:Advertising     FALSE      FALSE
CompPrice:Income       FALSE      FALSE
Price:Age              FALSE      FALSE
1 subsets of each size up to 5
Selection Algorithm: exhaustive
         ShelveLocGood ShelveLocMedium UrbanYes USYes CompPrice Income Advertising Population Price Age
1  ( 1 ) "*"           " "             " "      " "   " "       " "    " "         " "        " "   " "
2  ( 1 ) "*"           " "             " "      " "   " "       " "    " "         " "        "*"   " "
3  ( 1 ) "*"           " "             " "      " "   "*"       " "    " "         " "        "*"   " "
4  ( 1 ) "*"           " "             " "      " "   "*"       " "    "*"         " "        "*"   " "
5  ( 1 ) "*"           "*"             " "      " "   "*"       " "    "*"         " "        "*"   " "
         Education Income:Advertising CompPrice:Income Price:Age
1  ( 1 ) " "       " "                " "              " "
2  ( 1 ) " "       " "                " "              " "
3  ( 1 ) " "       " "                " "              " "
4  ( 1 ) " "       " "                " "              " "
5  ( 1 ) " "       " "                " "              " "
```

7.1 Variable Significance

Below we print the coefficients for the 5th model using the default model selection criteria. All coefficients are relatively small, as we would expect from the EDA above. This pretty much confirms what I would have guessed by looking at the data against sales. We still want to watch out for confounding between ‘Price’ and ‘CompPrice.’

HIDE

```
coef(interaction.lm.subsets, 5)
```

```
    (Intercept)   ShelveLocGood ShelveLocMedium       CompPrice     Advertising           Price
     -0.6995348       1.6746198       0.6277225       0.5090177       0.2832405      -0.7846476
```

7.2 Second Interaction Model

First, drop columns unneeded from analysis:

HIDE

```
scaled.merged.slim <- scaled.merged[ , -which(names(scaled.merged) %in% c("US","Urban"))]
```

A few hyper parameters we’d like to be consistent for all models

HIDE

```
nvmax <- 3
```

7.3 Forward Selection:

HIDE

```
interaction.subset.fwd <- regsubsets(Sales~. + Income*Advertising + Income*CompPrice + Income*Age,
                                      data=scaled.merged.slim,
                                      nvmax = nvmax,
                                      method="forward")
fwd.subset.summary <- summary(interaction.subset.fwd)
coef(interaction.subset.fwd, 1:nvmax)
```

```
[[1]]
  (Intercept) ShelveLocGood
    -0.259671      1.221981

[[2]]
  (Intercept) ShelveLocGood         Price
   -0.2708256     1.2744733    -0.4688868

[[3]]
  (Intercept) ShelveLocGood     CompPrice         Price
   -0.2709362     1.2749938     0.4932455    -0.7573701
```

7.4 Backward Selection:

This is really strange. I can't seem to find any documentation about this, but it appears that this model is actually 'forward.'

HIDE

```
interaction.subset.bk <- regsubsets(Sales~. + Income*Advertising + Income*CompPrice + Income*Age,
                                     data=scaled.merged.slim,
                                     nvmax = nvmax,
                                     method="backward")
bk.subset.summary <- summary(interaction.subset.bk)
coef(interaction.subset.bk, 1:nvmax)
```

```
[[1]]
  (Intercept) ShelveLocGood
    -0.259671      1.221981

[[2]]
  (Intercept) ShelveLocGood         Price
   -0.2708256     1.2744733    -0.4688868

[[3]]
  (Intercept) ShelveLocGood     CompPrice         Price
   -0.2709362     1.2749938     0.4932455    -0.7573701
```

7.5 Exhasutive

HIDE

```
interaction.subset.ex <- regsubsets(Sales~. + Income*Advertising + Income*CompPrice + Income*Age,
                                     data=scaled.merged.slim,
                                     nvmax = nvmax,
                                     method="exhaustive")
ex.subset.summary <- summary(interaction.subset.ex)
coef(interaction.subset.ex, 1:nvmax)
```

```
[[1]]
  (Intercept) ShelveLocGood 
    -0.259671      1.221981 

[[2]]
  (Intercept) ShelveLocGood         Price 
   -0.2708256     1.2744733    -0.4688868 

[[3]]
  (Intercept) ShelveLocGood     CompPrice         Price 
   -0.2709362     1.2749938     0.4932455    -0.7573701 
```

This is a list that may come in handy.

HIDE

```
model.list = list(list("Forward", interaction.subset.fwd, fwd.subset.summary),
                  list("Backward", interaction.subset.bk, bk.subset.summary),
                  list("Exhaustive", interaction.subset.ex, ex.subset.summary))
```

8 Evaluation Metric Plotting

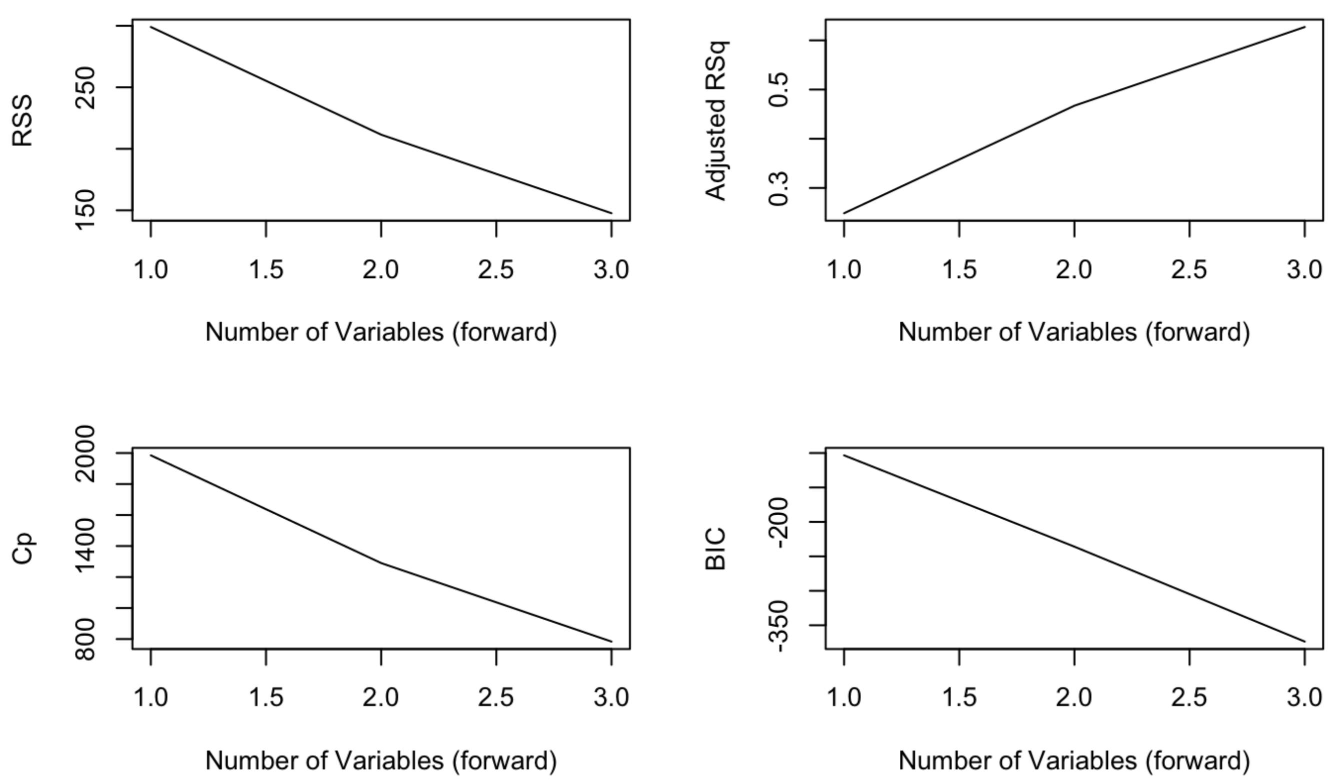
It is interesting to see that each model selected the same variables, in the same order.

HIDE

```
par(mfrow=c(2,2))
plot(fwd.subset.summary$rss,xlab="Number of Variables (forward)",ylab="RSS",type="l")
plot(fwd.subset.summary$adjr2,xlab="Number of Variables (forward)",ylab="Adjusted RSq",type="l")
```

HIDE

```
plot(fwd.subset.summary$cp,xlab="Number of Variables (forward)",ylab="Cp",type='l')
plot(fwd.subset.summary$bic,xlab="Number of Variables (forward)",ylab="BIC",type='l')
```

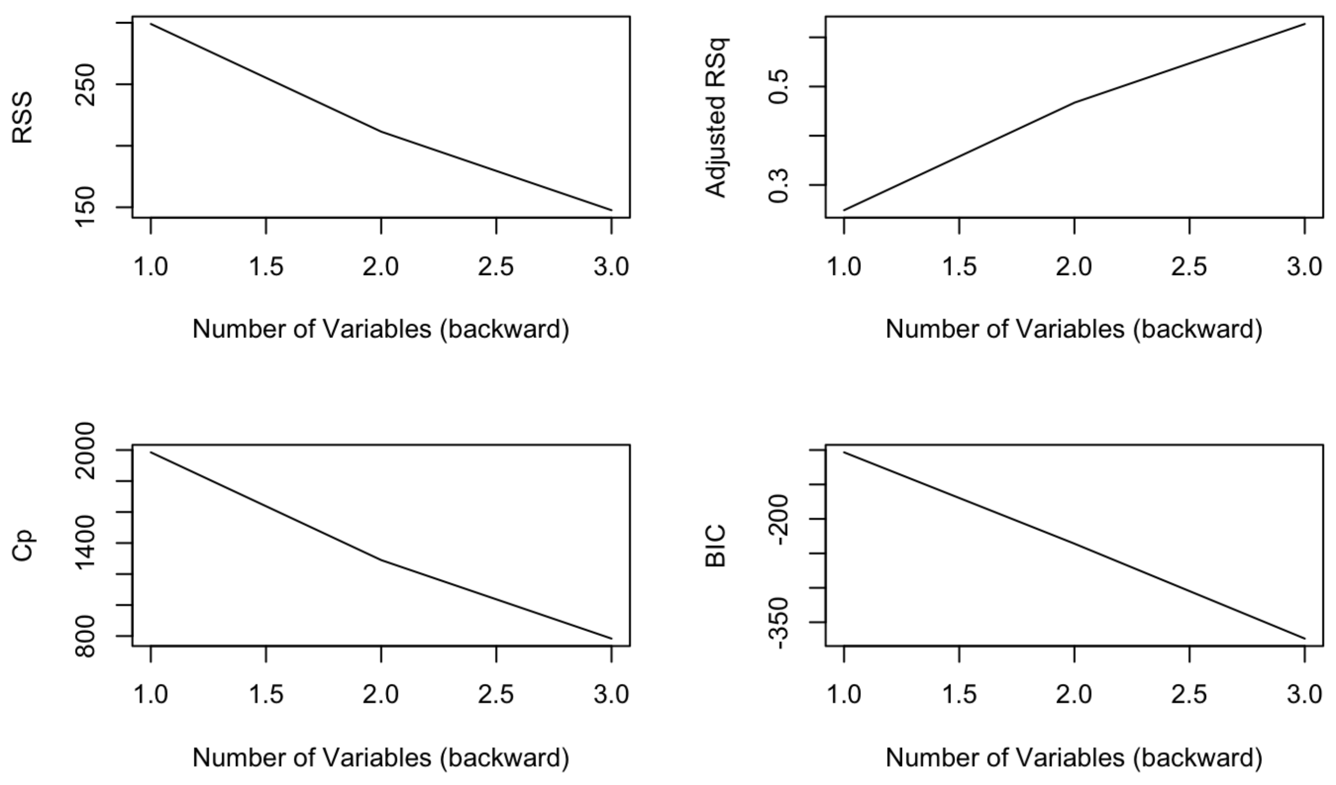


HIDE

```
par(mfrow=c(2,2))
plot(bk.subset.summary$rss,xlab="Number of Variables (backward)",ylab="RSS",type="l")
plot(bk.subset.summary$adjr2,xlab="Number of Variables (backward)",ylab="Adjusted RSq",type="l")
```

HIDE

```
plot(bk.subset.summary$cp,xlab="Number of Variables (backward)",ylab="Cp",type='l')
plot(bk.subset.summary$bic,xlab="Number of Variables (backward)",ylab="BIC",type='l')
```

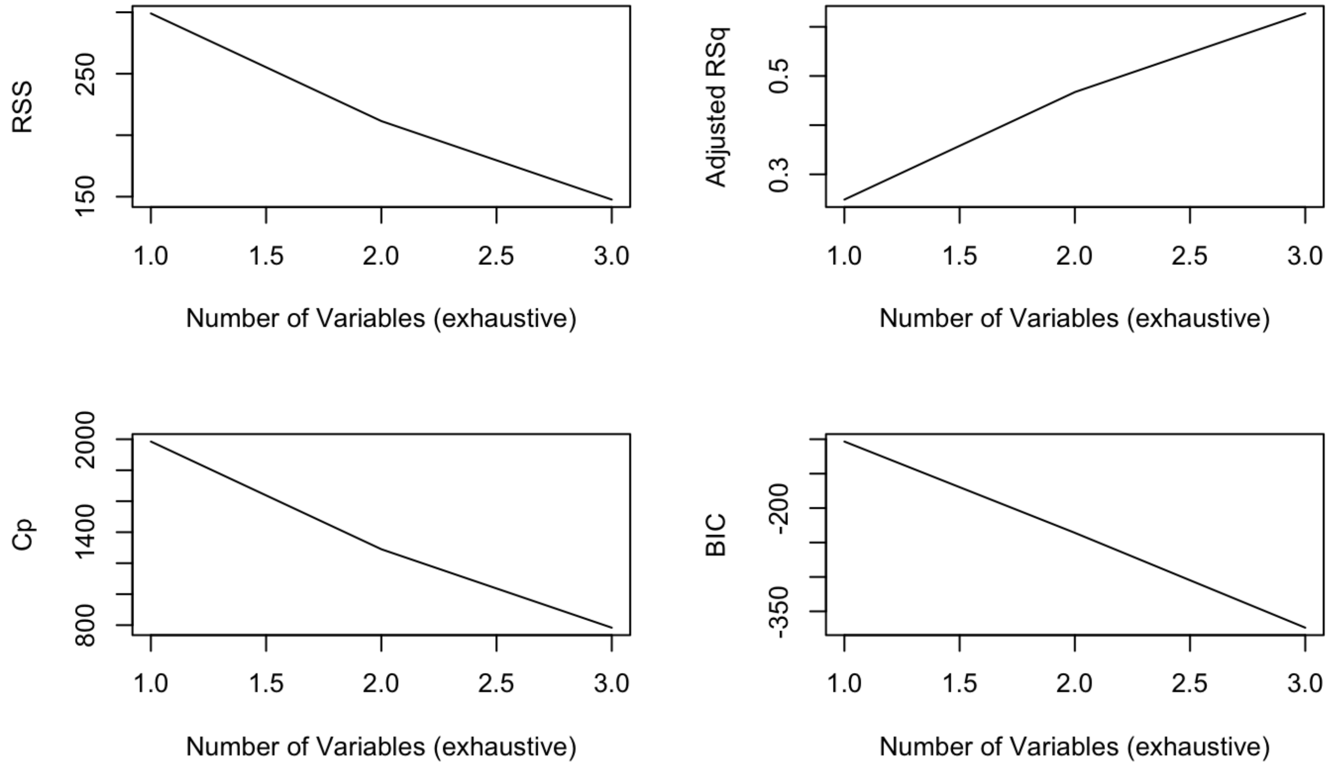


HIDE

```
par(mfrow=c(2,2))
plot(ex.subset.summary$rss,xlab="Number of Variables (exhaustive)",ylab="RSS",type="l")
plot(ex.subset.summary$adjr2,xlab="Number of Variables (exhaustive)",ylab="Adjusted RSq",type="l")
```

HIDE

```
plot(ex.subset.summary$cp,xlab="Number of Variables (exhaustive)",ylab="Cp",type='l')
plot(ex.subset.summary$bic,xlab="Number of Variables (exhaustive)",ylab="BIC",type='l')
```



9 Model Equations

Here we print the final equations for each model. Not, they are all the same.

HIDE

```
for (mod.obj in model.list) {
  mod.name <- mod.obj[[1]]
  best.bic <- min(mod.obj[[3]]$bic)
  mod.num <- which.min(mod.obj[[3]]$bic)
  mod.cc <- coef(mod.obj[[2]], mod.num)

  mod.equation.format <- paste("Y =", paste(round(mod.cc[1],2),
                      paste(round(mod.cc[-1],2),
                      names(mod.cc[-1]),
                      sep=" * ", collapse=" + "),
                      sep=" + "), "+ e")

  print(paste("Model Selection Method: ",mod.name))
  print(paste("Max BIC:", best.bic))
  print(paste("Model Number: ", mod.num))
  print(paste("Model Equation: ", mod.equation.format))
  print("")

}
```

```
[1] "Model Selection Method:  Forward"
[1] "Max BIC: -373.710213587368"
[1] "Model Number:  3"
[1] "Model Equation:  Y = -0.27 + 1.27 * ShelveLocGood + 0.49 * CompPrice + -0.76 * Price + e"
[1] ""
[1] "Model Selection Method:  Backward"
[1] "Max BIC: -373.710213587368"
[1] "Model Number:  3"
[1] "Model Equation:  Y = -0.27 + 1.27 * ShelveLocGood + 0.49 * CompPrice + -0.76 * Price + e"
[1] ""
[1] "Model Selection Method:  Exhaustive"
[1] "Max BIC: -373.710213587368"
[1] "Model Number:  3"
[1] "Model Equation:  Y = -0.27 + 1.27 * ShelveLocGood + 0.49 * CompPrice + -0.76 * Price + e"
[1] ""
```