Real_Case_Data_Processing_and_Machine_Learning_in_R

Brian Han

"The best thing about being a statistician is that you get to play in everyone's backyard" – John Tukey

John Tukey was a Professor of Statistics at Princeton University. He is best known for development of the FFT algorithm and box plot!

About this course

Prerequisites: linear algebra, random variables, expectation, variance

What you will learn in 2 hours:

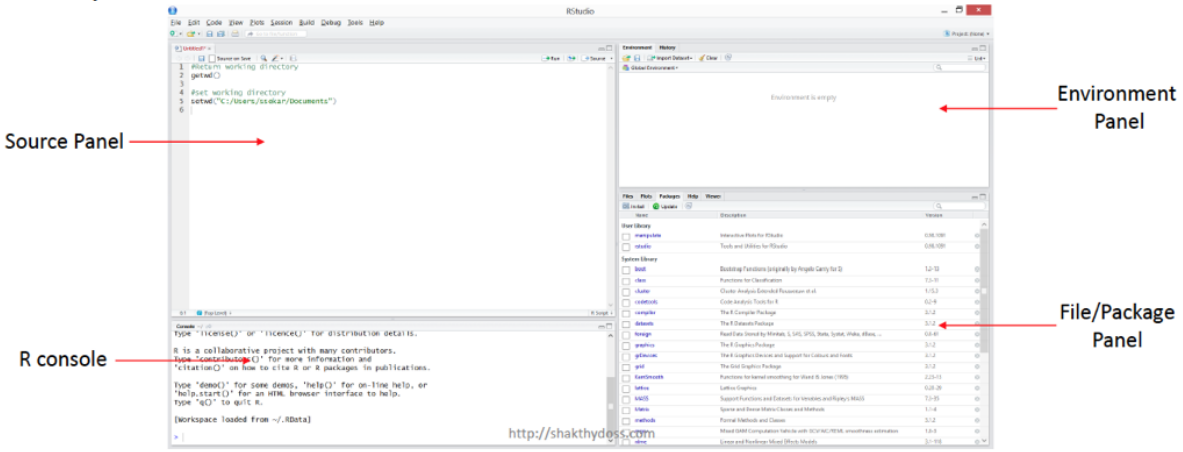
1. R Programming:
    - Install R and RStudio
    - Data Types, Matrix, List, Data Frame
2. Data Exploration:
    - Data Cleaning
    - Summary Statistics
    - Vishalizing Data
3. Building Statistical Learning Models in r
    - Variable Selection and Shrinkage
    - Tree Based Model

R Programming

Installation

- One of the best things about R is that it is free and has well supported IDE. You can install R at https://cran.r-project.org. RStudio is the mostly used IDE for R that is avaiable in https://www.rstudio.com/products/rstudio/download/.

Open RStudio



-

You can create a new script in the Source Panel and save it. Now you are ready to play with it! Usually, the first thing that I do is clean everything in the Enviroment Panel and change the directory if necessary. For example

```
rm(list=ls())
dir <- getwd()
setwd(dir)  # you can change dir
```

cmd + enter or *control + enter* is a short key for running the code of current line. "#" is used for comments.

- In RStudio, we can install packages in File/Package Panel or by entering

```
#install.packages("ggplot2")
```

```
#library("ggplot2")
```

*Getting helps

```
help(c)
?c
```

R 101

```
library("ggplot2")

#Vectors Windows/Linux: "Alt" + "-"  Mac: "Option" + "-" to get <-
x <- c(0.5, 0.6) ## numeric
x <- c(TRUE, FALSE) ## logical
x <- c(T, F) ## logical
x <- c("a", "b", "c") ## character
x <- 9:29 ## integer
x <- c(1+0i, 2+4i) ## complex
class(x)
```

```
## [1] "complex"
```

```r
#Matrix
m <- matrix(nrow = 2, ncol = 3)
m
```

```
##      [,1] [,2] [,3]
## [1,]   NA   NA   NA
## [2,]   NA   NA   NA
```

```r
dim(m)
```

```
## [1] 2 3
```

```r
ncol(m)
```

```
## [1] 3
```

```r
nrow(m)
```

```
## [1] 2
```

```r
#Factor
x <- factor(c("yes", "yes", "no", "yes", "no"))
x
```

```
## [1] yes yes no  yes no
## Levels: no yes
```

```r
table(x)
```

```
## x
##  no yes
##   2   3
```

```r
unclass(x)
```

```
## [1] 2 2 1 2 1
## attr(,"levels")
## [1] "no"  "yes"
```

```r
#Combine
x <- c(1:3)
y <- 10:12
cbind(x, y)
```

```
##      x  y
## [1,] 1 10
## [2,] 2 11
## [3,] 3 12
```

```r
rbind(x, y)
```

```
##   [,1] [,2] [,3]
## x    1    2    3
## y   10   11   12
```

```r
#List
x <- list(1, "a", TRUE, 1 + 4i)
x
```

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

```
## [[2]]
## [1] "a"
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] 1+4i
```

```
#DataFrame
x <- data.frame(foo = 1:4, bar = c(NA, T, F, F))
names(x)
```

```
## [1] "foo" "bar"
```

```
x[!is.na(x)] <- 0


#Operations
x <- c(1:3)
y <- 10:12
x+y
```

```
## [1] 11 13 15
```

```
x/y
```

```
## [1] 0.1000000 0.1818182 0.2500000
```

```
t(x)%*%y
```

```
##      [,1]
## [1,]   68
```

```
x>=y
```

```
## [1] FALSE FALSE FALSE
```

```
#loops
x <- c("a", "b", "c", "d")
for(i in 1:4) {
  print(x[i])
}
```

```
## [1] "a"
## [1] "b"
## [1] "c"
## [1] "d"
```

```
for(i in seq_along(x)) {
  print(x[i])
}
```

```
## [1] "a"
## [1] "b"
## [1] "c"
## [1] "d"
```

```
for(letter in x) {
  print(letter)
```

```
}
```

```
## [1] "a"
## [1] "b"
## [1] "c"
## [1] "d"
```

```
for(i in 1:4) print(x[i])
```

```
## [1] "a"
## [1] "b"
## [1] "c"
## [1] "d"
```

- Summary

1. atomic classes
2. vector, list
3. factor
4. missing values
5. dataframe, names
6. basic operations
7. loops

Data

- Common data types includes: Numeric, Integer, Logical and Character

```
x = c(2, 3, 5)             #Numeric
y = c("aa", "bb", "cc")    #Character
z = c(TRUE, FALSE, TRUE)   #Logical
```

- Matrix

```
A = matrix(
  c(2, 4, 3, 1, 5, 7),  # the data elements
   nrow=2,              # number of rows
   ncol=3,              # number of columns
   byrow = TRUE)        # fill matrix by rows
A
```

```
##      [,1] [,2] [,3]
## [1,]    2    4    3
## [2,]    1    5    7
```

```
dimnames(A) = list(
  c("row1", "row2"),         # row names
   c("col1", "col2", "col3")) # column names

A                 # print A
```

```
##      col1 col2 col3
## row1    2    4    3
## row2    1    5    7
```

```
A["row2", "col3"] # element at 2nd row, 3rd column
```

```
## [1] 7
```

- DataFrame

```
#View(mtcars) # First look at the data
dim(mtcars)  # dimension of this big matrix
```

```
## [1] 32 11
```

```
nrow(mtcars) # number of observation
```

```
## [1] 32
```

```
names(mtcars)
```

```
##  [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
## [11] "carb"
```

Data Exploration

Data Cleaning

```
if(!file.exists("./data")){dir.create("./data")}
#fileurl <- "https://data.baltimorecity.gov/resource/6act-qzuy.csv"
#download.file(fileurl, destfile = "./data/hosp.csv",method = "curl")
#https://data.baltimorecity.gov/Financial/Real-Property-Taxes/27w9-urtv
raw_data <- read.csv("./data/tax.csv")


str(raw_data)
```

```
## 'data.frame':    238278 obs. of  16 variables:
##  $ PropertyID     : Factor w/ 238278 levels "0001 001 ","0001 002 ",..: 32458 113982 35423 139675 354
##  $ Block          : Factor w/ 5558 levels "0001 ","0002 ",..: 707 2438 763 2996 763 1 1 1 1 2438 ...
##  $ Lot            : Factor w/ 3053 levels "001 ","001A",..: 976 986 350 1015 489 507 525 552 584 1015
##  $ Ward           : int  23 13 23 8 23 15 15 15 15 13 ...
##  $ Sect           : int  40 30 90 240 90 370 370 370 370 30 ...
##  $ PropertyAddress: Factor w/ 233461 levels "                                    ",..: 9882 132203 55995
##  $ LotSize        : Factor w/ 60352 levels "                 ",..: 17988 5557 8855 13610 7457 7718 77
##  $ CityTax        : Factor w/ 17101 levels "","$0.74","$1.51",..: 9220 6888 11899 4717 13305 10555 10
##  $ StateTax       : Factor w/ 16959 levels "","$0.04","$0.08",..: 3982 1683 6652 16488 8058 5159 5159
##  $ ResCode        : Factor w/ 2 levels "NOT A PRINCIPAL RESIDENCE",..: 1 1 1 1 1 1 2 1 1 2 ...
##  $ AmountDue      : Factor w/ 93462 levels "","$0.01","$0.02",..: 32706 53099 52748 1 83228 1 1 1 1 5
##  $ AsOfDate       : Factor w/ 266 levels "01/03/2017","01/05/2017",..: 206 109 150 236 136 31 31 31 3
##  $ Neighborhood   : Factor w/ 282 levels ""," ","ABELL",..: 85 109 238 14 238 73 73 73 73 109 ...
##  $ PoliceDistrict : Factor w/ 12 levels ""," ","CENTRAL",..: 10 7 10 5 10 12 12 12 12 7 ...
##  $ CouncilDistrict: int  11 7 11 13 11 7 7 7 7 7 ...
##  $ Location       : Factor w/ 222130 levels "","(39.19977274, -76.55031677)",..: 21808 152182 12483 1
```

```
names(raw_data)
```

```
##  [1] "PropertyID"      "Block"           "Lot"
##  [4] "Ward"            "Sect"            "PropertyAddress"
##  [7] "LotSize"         "CityTax"         "StateTax"
## [10] "ResCode"         "AmountDue"       "AsOfDate"
## [13] "Neighborhood"    "PoliceDistrict"  "CouncilDistrict"
## [16] "Location"
```

```r
#select
head(select(raw_data,1:5))
```

```
##   PropertyID Block  Lot Ward Sect
## 1  0950 052  0950  052    23   40
## 2  3523 053  3523  053    13   30
## 3  1029 016  1029  016    23   90
## 4  4154 055  4154  055     8  240
## 5  1029 022  1029  022    23   90
## 6  0001 023  0001  023    15  370
```

```r
head(select(raw_data, Lot: CityTax))
```

```
##    Lot Ward Sect                   PropertyAddress           LotSize
## 1 052    23   40 1105 S CHARLES ST                 16X50
## 2 053    13   30 3535 BUENA VISTA AVE              12-1X175
## 3 016    23   90 1830 S CHARLES ST                 13-9X123
## 4 055     8  240 2829 LAKE AVE                    15-3X106
## 5 022    23   90 1837 S HANOVER ST                 12X71
## 6 023    15  370 1816 N PAYSON ST                 13-10X80
##    CityTax
## 1 $3929.50
## 2 $2697.60
## 3 $5604.26
## 4 $1852.35
## 5 $6618.11
## 6  $472.08
```

```r
head(select(raw_data, Lot: CityTax, Neighborhood:Location ))
```

```
##    Lot Ward Sect                   PropertyAddress           LotSize
## 1 052    23   40 1105 S CHARLES ST                 16X50
## 2 053    13   30 3535 BUENA VISTA AVE              12-1X175
## 3 016    23   90 1830 S CHARLES ST                 13-9X123
## 4 055     8  240 2829 LAKE AVE                    15-3X106
## 5 022    23   90 1837 S HANOVER ST                 12X71
## 6 023    15  370 1816 N PAYSON ST                 13-10X80
##    CityTax    Neighborhood PoliceDistrict CouncilDistrict
## 1 $3929.50    FEDERAL HILL       SOUTHERN               11
## 2 $2697.60         HAMPDEN       NORTHERN                7
## 3 $5604.26 SOUTH BALTIMORE       SOUTHERN               11
## 4 $1852.35   BELAIR-EDISON   NORTHEASTERN               13
## 5 $6618.11 SOUTH BALTIMORE       SOUTHERN               11
## 6  $472.08      EASTERWOOD        WESTERN                7
##                       Location
## 1 (39.27650571, -76.61413277)
## 2 (39.32926771, -76.63821365)
## 3 (39.26862331, -76.61415384)
## 4 (39.32341678, -76.57362347)
## 5 (39.26874293, -76.61476111)
## 6  (39.30913416, -76.6501138)
```

```r
head(select(raw_data, starts_with("A") ))
```

```
##   AmountDue   AsOfDate
## 1    $19.70 10/03/2017
```

```
## 2  $2818.51 07/01/2017
## 3    $28.02 08/10/2017
## 4           10/28/2017
## 5  $6947.84 07/31/2017
## 6           03/29/2016
```

```
#filter
df <- filter(raw_data, Ward >=8)
head(select(df,1:3))
```

```
##   PropertyID Block  Lot
## 1  0950 052  0950  052
## 2  3523 053  3523  053
## 3  1029 016  1029  016
## 4  4154 055  4154  055
## 5  1029 022  1029  022
## 6  0001 023  0001  023
```

```
df <- filter(raw_data, Ward >=8 & Ward <= 10)
head(select(df,1:3))
```

```
##   PropertyID Block  Lot
## 1  4154 055  4154  055
## 2  4154 076  4154  076
## 3  4155 006  4155  006
## 4  4155 031  4155  031
## 5  4155 033  4155  033
## 6  4155 034  4155  034
```

```
#arrange
df <- arrange(raw_data, Lot)
head(select(df, 1:3))
```

```
##   PropertyID Block  Lot
## 1  5216A001  5216A 001
## 2  0002 001  0002  001
## 3  1030 001  1030  001
## 4  7437 001  7437  001
## 5  1037 001  1037  001
## 6  1038 001  1038  001
```

```
tail(select(df, 1:3))
```

```
##        PropertyID Block  Lot
## 238273  3702 946  3702  946
## 238274  3702 947  3702  947
## 238275  3702 948  3702  948
## 238276  3702 949  3702  949
## 238277  3702 950  3702  950
## 238278  3702 951  3702  951
```

```
df <- arrange(raw_data, desc(Lot) )
head(select(df, 1:3))
```

```
##   PropertyID Block  Lot
## 1  3702 951  3702  951
## 2  3702 950  3702  950
## 3  3702 949  3702  949
```

```
## 4  3702 948  3702  948
## 5  3702 947  3702  947
## 6  3702 946  3702  946
```

```
tail(select(df, 1:3))
```

```
##        PropertyID Block  Lot
## 238273  8280 001  8280  001
## 238274  8286C001  8286C 001
## 238275  8421 001  8421  001
## 238276  8470 001  8470  001
## 238277  8434F001  8434F 001
## 238278  8486 001  8486  001
```

```
#rename
names(raw_data)
```

```
##  [1] "PropertyID"      "Block"           "Lot"
##  [4] "Ward"            "Sect"            "PropertyAddress"
##  [7] "LotSize"         "CityTax"         "StateTax"
## [10] "ResCode"         "AmountDue"       "AsOfDate"
## [13] "Neighborhood"    "PoliceDistrict"  "CouncilDistrict"
## [16] "Location"
```

```
df <- rename(raw_data, ID = PropertyID )
head(select(df, 1:3))
```

```
##          ID Block  Lot
## 1 0950 052  0950  052
## 2 3523 053  3523  053
## 3 1029 016  1029  016
## 4 4154 055  4154  055
## 5 1029 022  1029  022
## 6 0001 023  0001  023
```

```
#mutate
df <-  mutate(raw_data, city_tax = as.numeric(gsub('[$]', '', CityTax)))
df <- mutate(df,  city_tax_nor =  (city_tax - mean(city_tax, na.rm = T))/ sd(city_tax, na.rm = T)  )

#group_by and summarize
df <-  mutate(raw_data, city_tax = as.numeric(gsub('[$]', '', CityTax)))
group_df <- group_by(df, Ward)
df <- summarise(group_df,
                mean_citytax = mean(city_tax, na.rm = T),
                sd_citytax = mean(city_tax, na.rm = T))

#Chaining
df <- raw_data %>%
  select(PropertyID, Block, Lot, Ward, 8:9) %>%
  rename(ID = PropertyID) %>%
  filter(!is.na(CityTax)& CityTax != "") %>%
  arrange(desc(Ward)) %>%
  mutate(city_tax = as.numeric(gsub('[$]', '', CityTax))) %>%
  group_by(Ward) %>%
  summarise(mean_citytax = mean(city_tax),
            sd_citytax = sd(city_tax))
```

Summary Statistics

- Data type

```
str(mtcars)
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

```
df <- mtcars
```

Let's convert some of them to factor variables

```
factorname <- c("cyl", "vs", "am", "gear", "carb")
df[factorname] <- lapply(df[factorname], as.factor)
str(df)
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : Factor w/ 3 levels "4","6","8": 2 2 1 2 3 2 3 1 1 2 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : Factor w/ 2 levels "0","1": 1 1 2 2 1 2 1 2 2 2 ...
##  $ am  : Factor w/ 2 levels "0","1": 2 2 2 1 1 1 1 1 1 1 ...
##  $ gear: Factor w/ 3 levels "3","4","5": 2 2 2 1 1 1 1 2 2 2 ...
##  $ carb: Factor w/ 6 levels "1","2","3","4",..: 4 4 1 1 2 1 4 2 2 4 ...
```

- Missing Values

```
sum(is.na(df))
```

```
## [1] 0
```

- Summary Satistics

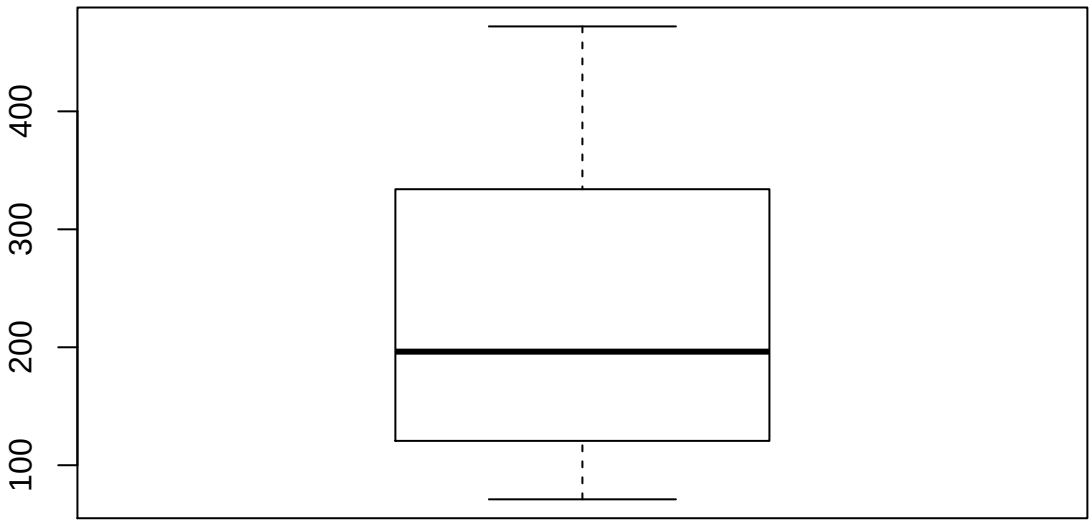
```
summary(df)
```

```
##       mpg        cyl         disp             hp             drat
##  Min.   :10.40   4:11   Min.   : 71.1   Min.   : 52.0   Min.   :2.760
##  1st Qu.:15.43   6: 7   1st Qu.:120.8   1st Qu.: 96.5   1st Qu.:3.080
##  Median :19.20   8:14   Median :196.3   Median :123.0   Median :3.695
##  Mean   :20.09          Mean   :230.7   Mean   :146.7   Mean   :3.597
##  3rd Qu.:22.80          3rd Qu.:326.0   3rd Qu.:180.0   3rd Qu.:3.920
##  Max.   :33.90          Max.   :472.0   Max.   :335.0   Max.   :4.930
##        wt             qsec       vs     am     gear   carb
##  Min.   :1.513   Min.   :14.50   0:18   0:19   3:15   1: 7
```

```
##  1st Qu.:2.581   1st Qu.:16.89   1:14   1:13   4:12   2:10
##  Median :3.325   Median :17.71                 5: 5   3: 3
##  Mean   :3.217   Mean   :17.85                        4:10
##  3rd Qu.:3.610   3rd Qu.:18.90                        6: 1
##  Max.   :5.424   Max.   :22.90                        8: 1
```

- BoxPlot

```
boxplot(df$disp)
```




```
quantile(df$disp, c(.25, .75))
```

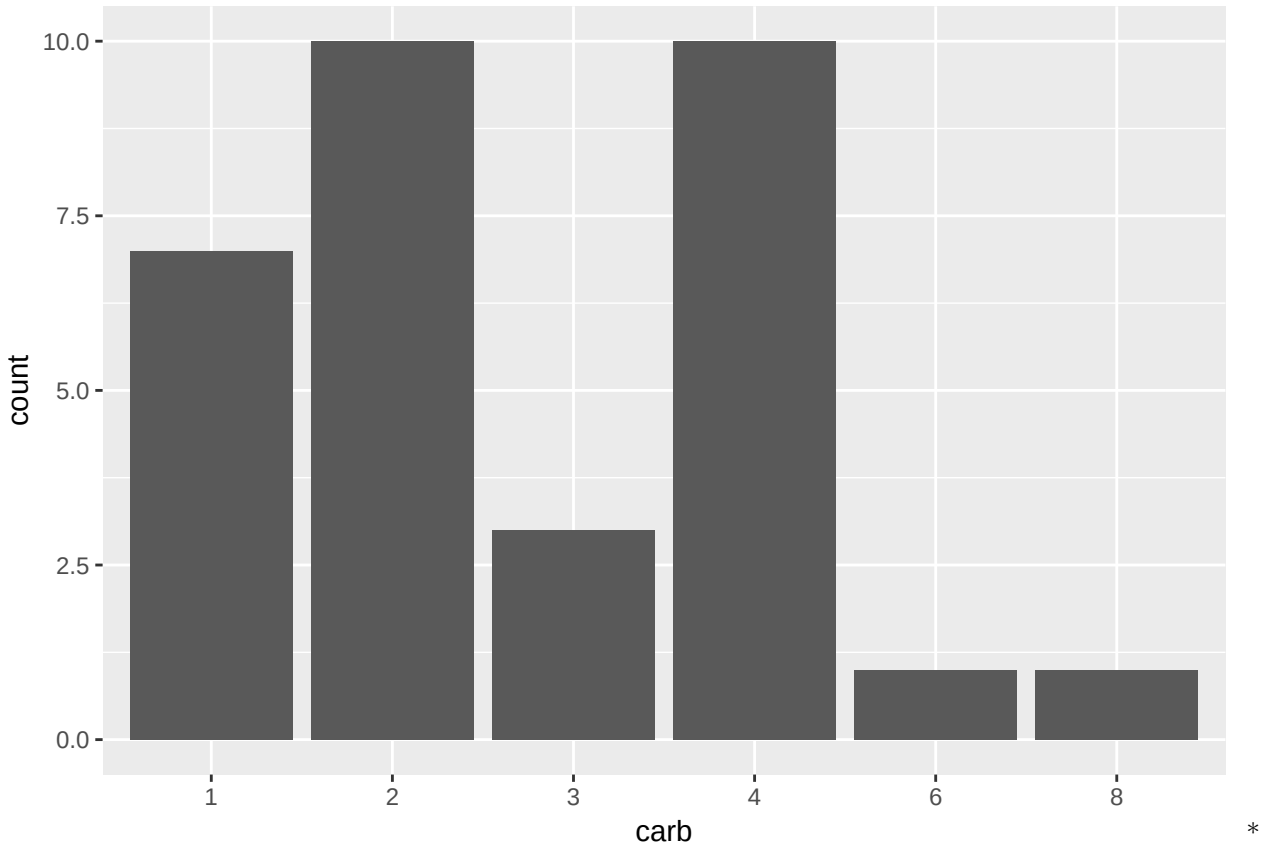
```
##     25%     75%
## 120.825 326.000
```

The lengths of the box is called Interquartile Range (IQR)

Visualizing Data

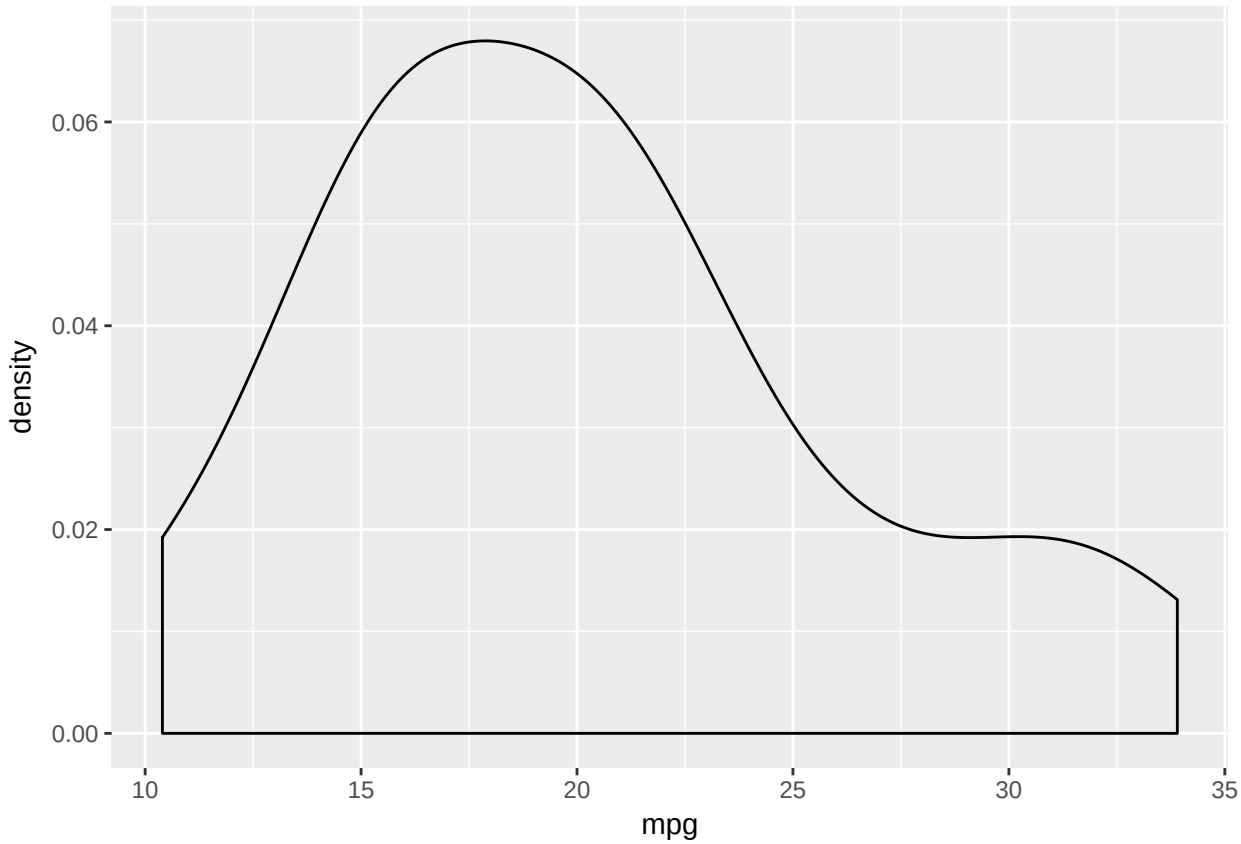
We are going to one of the best packages in R – ggplot2. Here is a cheatsheet for starters https://www.rstudio.com/wp-content/uploads/2016/11/ggplot2-cheatsheet-2.1.pdf * Discrete For example, let's look at variable carb

```
ggplot(df, aes(x = carb)) +
  geom_bar()
```



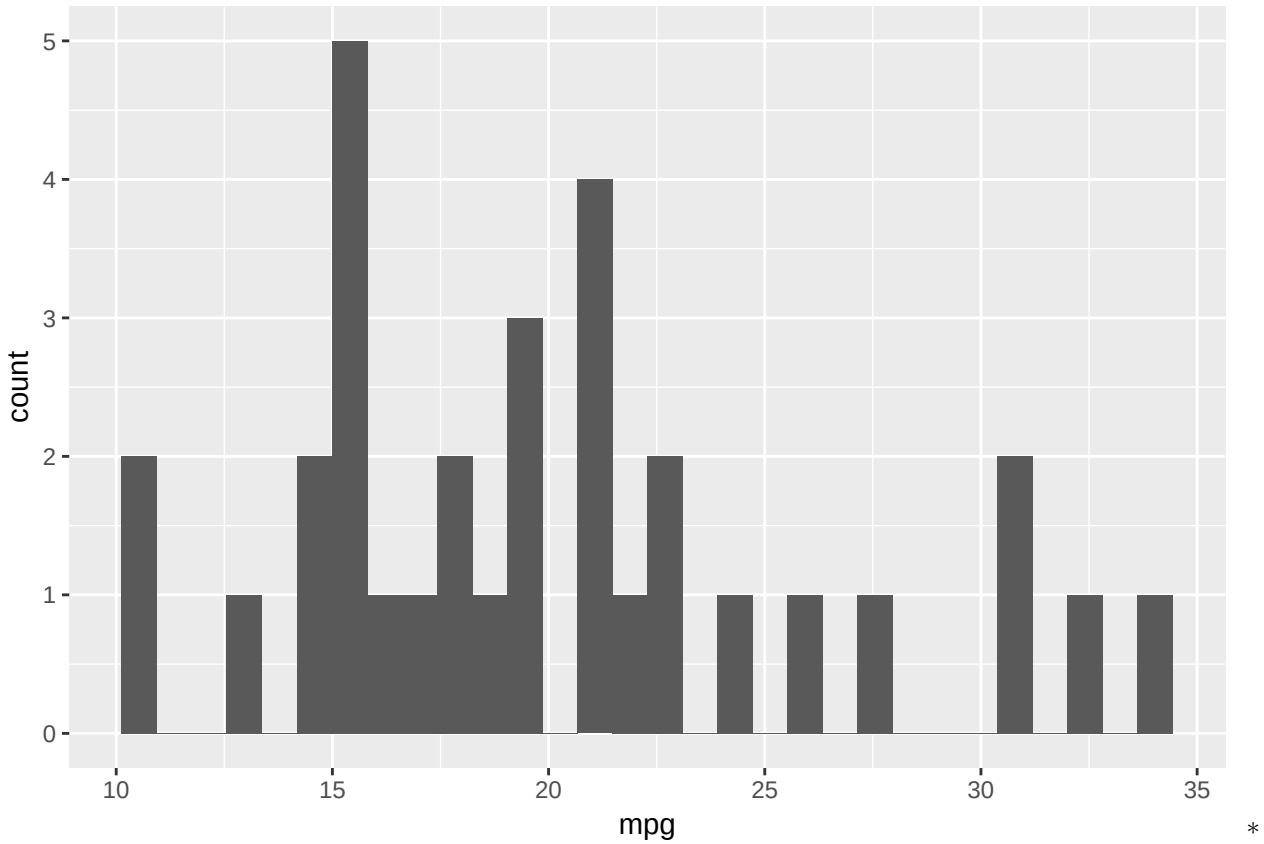
Continuous

```
a <- ggplot(df, aes(x = mpg))
a + geom_density()
```



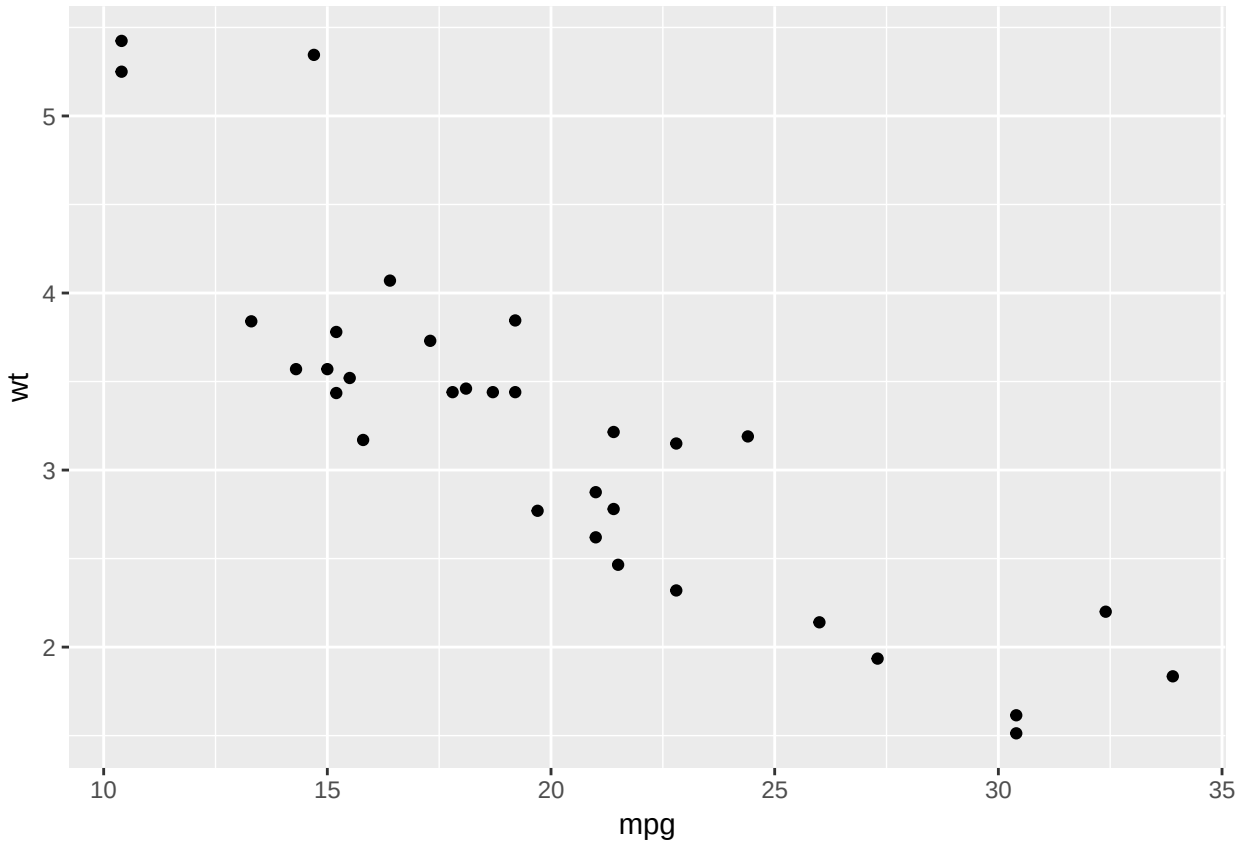

```
a + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



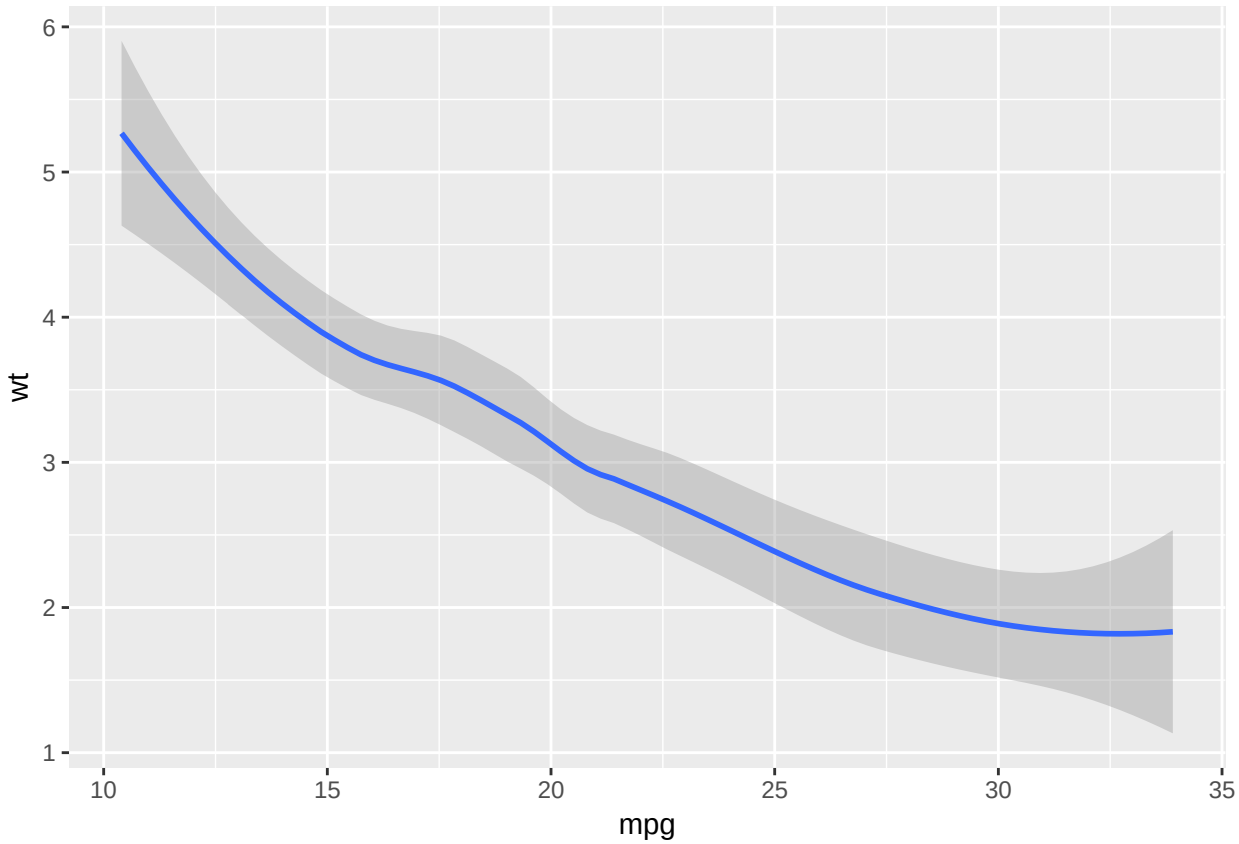
Continuous x + Continuous y

```
a <- ggplot(df, aes(x = mpg, y= wt ))
a + geom_point()
```

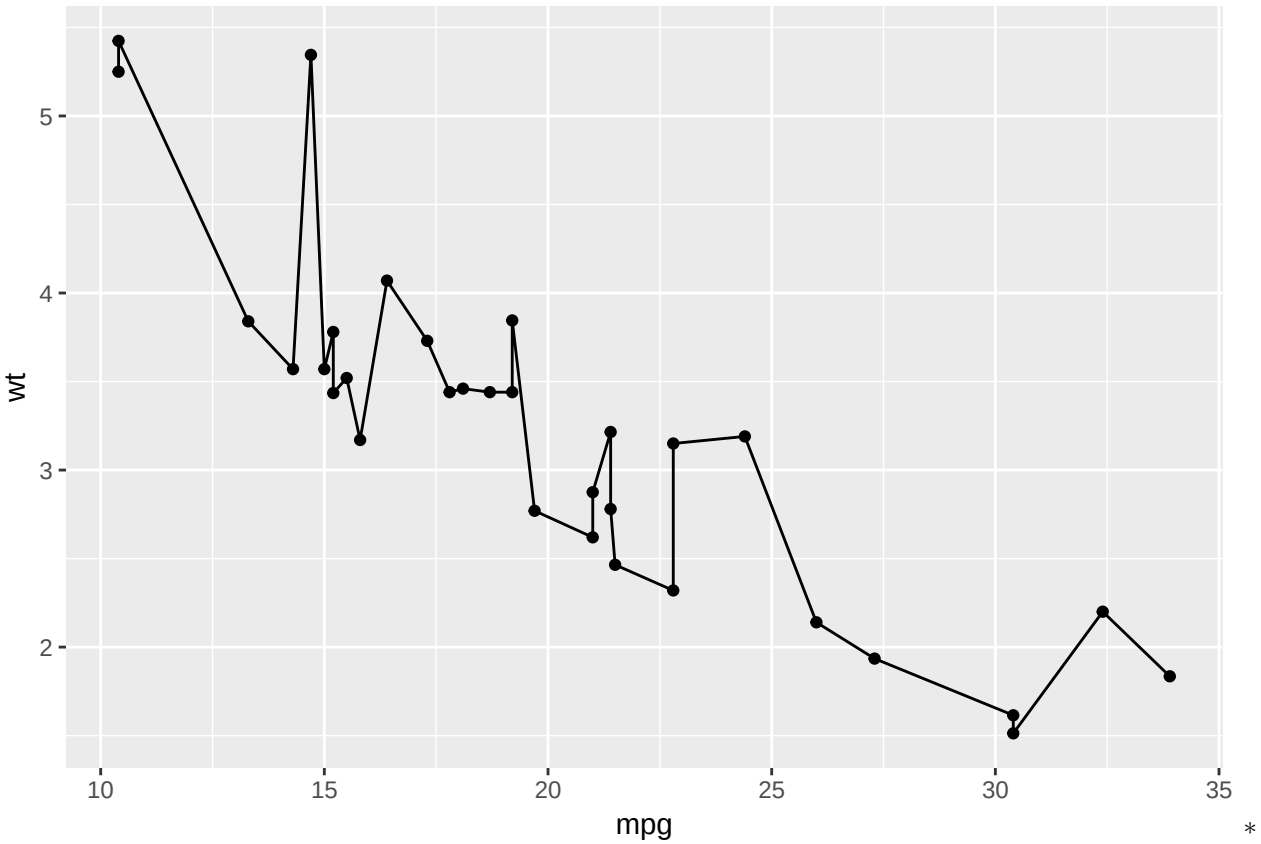



```
a + geom_smooth()
```

```
## `geom_smooth()` using method = 'loess'
```

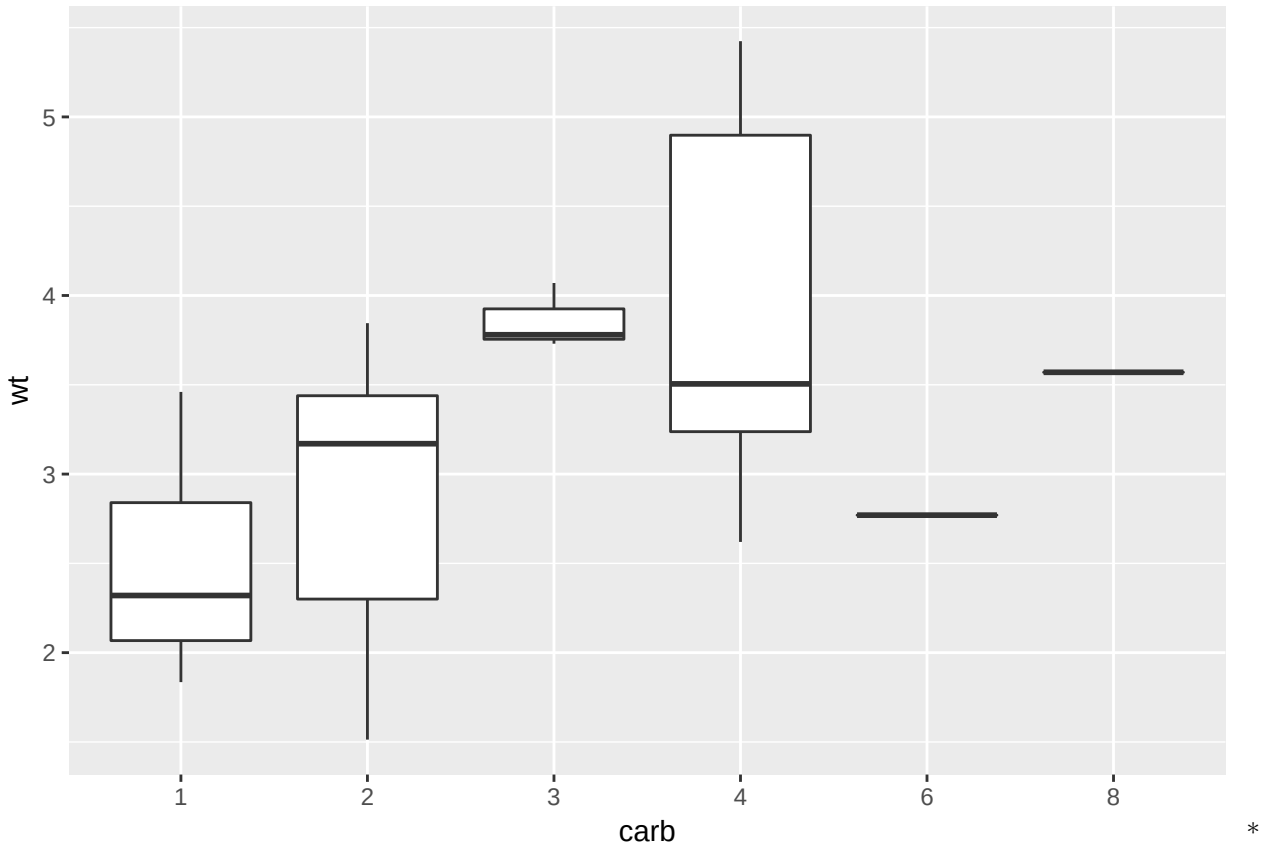



```
a + geom_line() + geom_point()
```



Discrete x + Continous y

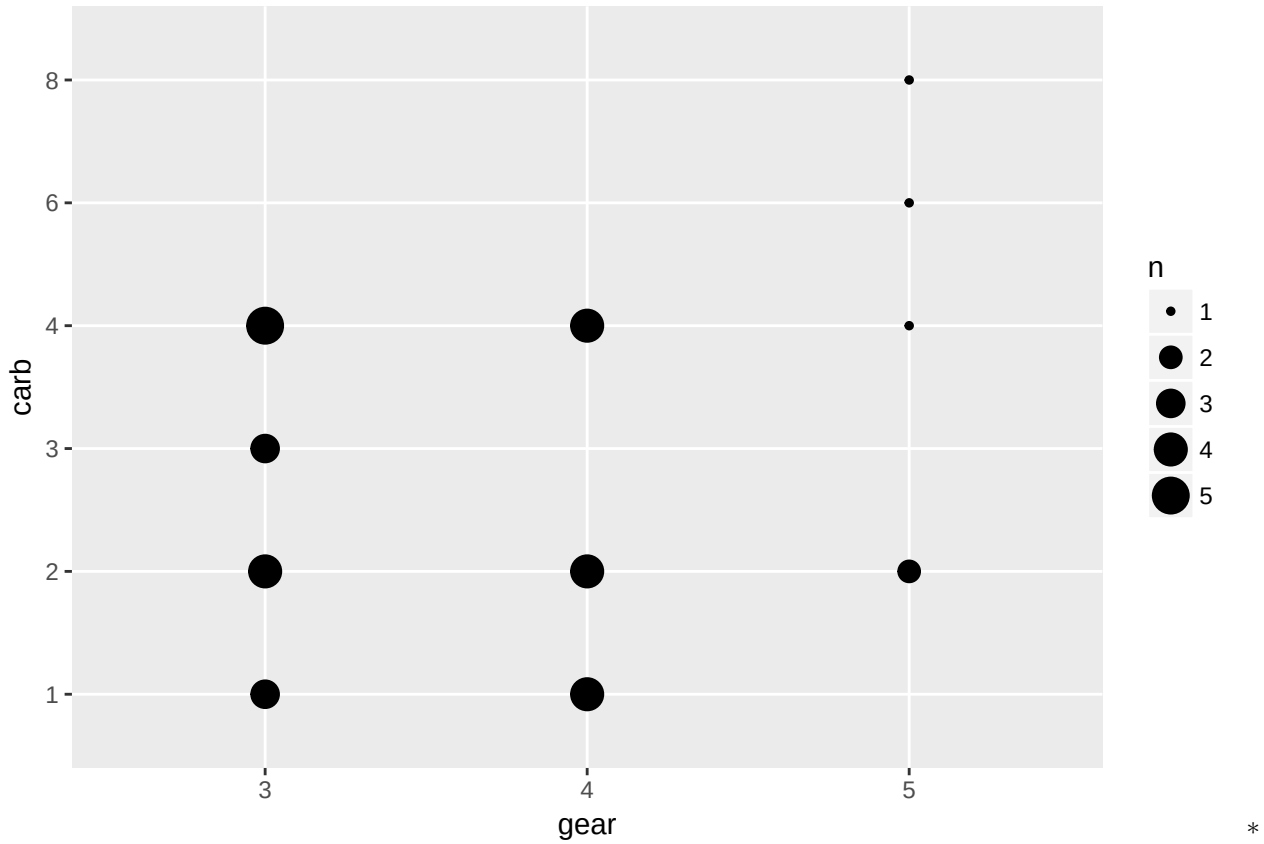
```
a <- ggplot(df, aes(x = carb, y = wt))
a+ geom_boxplot()
```



*

Discrete x + Discrete y

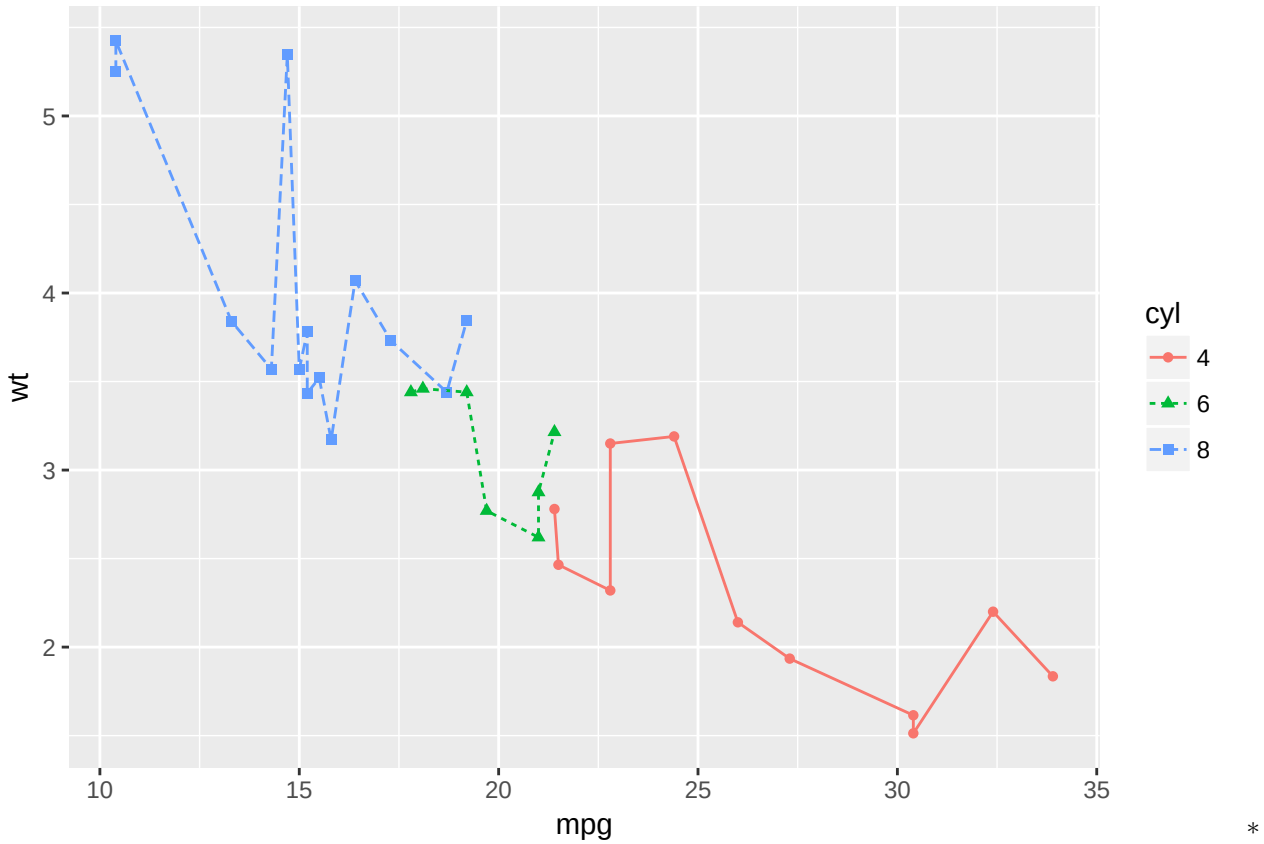
```
ggplot(df, aes(gear, carb)) + geom_count()
```



*

By group

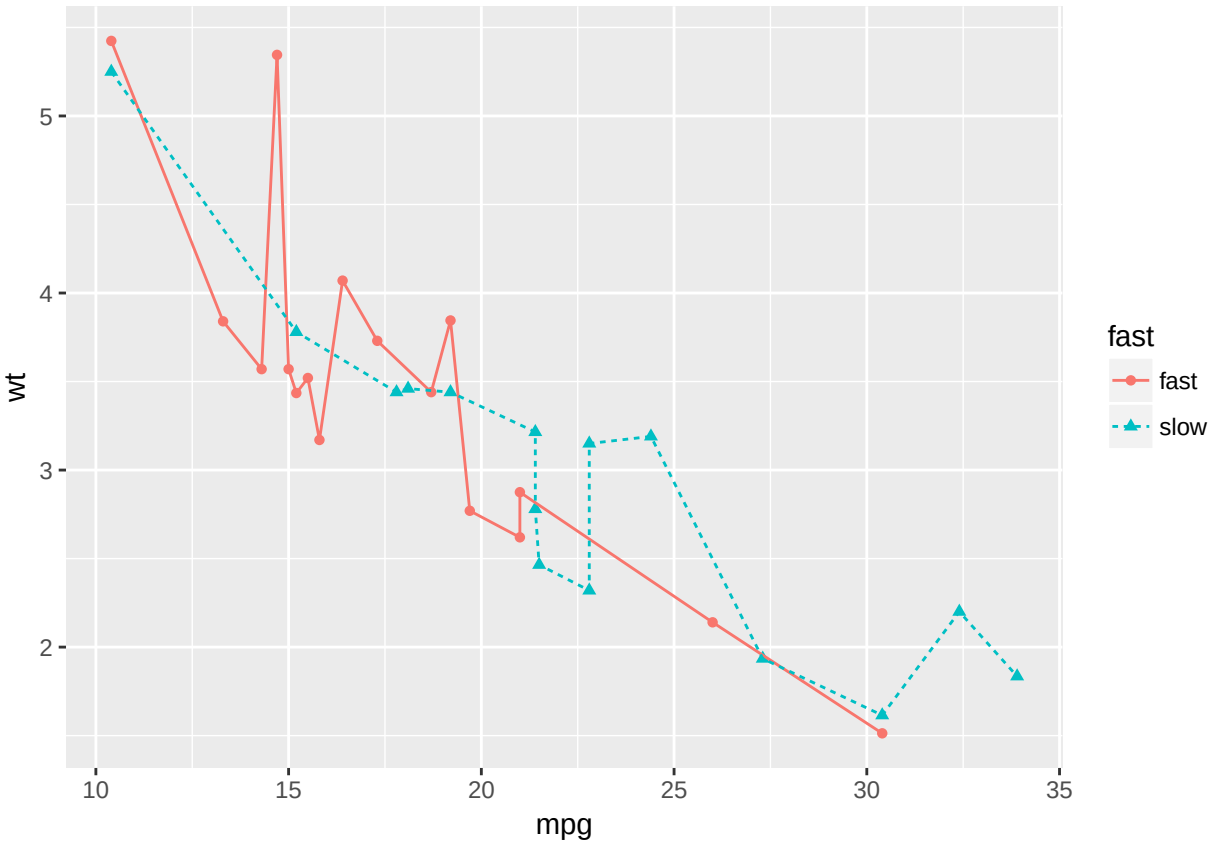
```
ggplot(df, aes(mpg, wt, linetype = cyl, color = cyl, shape = cyl)) +
  geom_point() +
  geom_line()
```



*

Combine ddplyr and ggplot

```
df %>%
  mutate(fast = ifelse(qsec >= mean(qsec), "slow", "fast")) %>%
  ggplot( aes(mpg, wt, linetype = fast , color = fast, shape = fast )) +
  geom_point() +
  geom_line()
```



Building Models in r

Variable Selection and Shrinkage

```
##########Variable Selection
library(ISLR)
df <- Hitters

#delete na
names(df)
```

```
##  [1] "AtBat"     "Hits"      "HmRun"     "Runs"      "RBI"
##  [6] "Walks"     "Years"     "CAtBat"    "CHits"     "CHmRun"
## [11] "CRuns"     "CRBI"      "CWalks"    "League"    "Division"
## [16] "PutOuts"   "Assists"   "Errors"    "Salary"    "NewLeague"
```

```
dim(df)
```

```
## [1] 322  20
```

```
sum(is.na(df$Salary))
```

```
## [1] 59
```

```
df <- na.omit(df)
dim(df)
```

```
## [1] 263  20
```

```r
sum(is.na(df))
```

```
## [1] 0
```

```r
#Best Subset Selection
library(leaps)
fit <- regsubsets(Salary~., df)
summary(fit)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., df)
## 19 Variables  (and intercept)
##            Forced in Forced out
## AtBat          FALSE      FALSE
## Hits           FALSE      FALSE
## HmRun          FALSE      FALSE
## Runs           FALSE      FALSE
## RBI            FALSE      FALSE
## Walks          FALSE      FALSE
## Years          FALSE      FALSE
## CAtBat         FALSE      FALSE
## CHits          FALSE      FALSE
## CHmRun         FALSE      FALSE
## CRuns          FALSE      FALSE
## CRBI           FALSE      FALSE
## CWalks         FALSE      FALSE
## LeagueN        FALSE      FALSE
## DivisionW      FALSE      FALSE
## PutOuts        FALSE      FALSE
## Assists        FALSE      FALSE
## Errors         FALSE      FALSE
## NewLeagueN     FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns
## 1  ( 1 ) " "   " "  " "   " "  " " " "   " "   " "    " "   " "    " "
## 2  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "
## 3  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "
## 4  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "
## 5  ( 1 ) "*"   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "
## 6  ( 1 ) "*"   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "
## 7  ( 1 ) " "   "*"  " "   " "  " " " "   " "   "*"    "*"   "*"    " "
## 8  ( 1 ) "*"   "*"  " "   " "  " " " "   " "   " "    " "   "*"    "*"
##          CRBI CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 ) "*"  " "    " "     " "       " "     " "     " "    " "
## 2  ( 1 ) "*"  " "    " "     " "       " "     " "     " "    " "
## 3  ( 1 ) "*"  " "    " "     " "       "*"     " "     " "    " "
## 4  ( 1 ) "*"  " "    " "     "*"       "*"     " "     " "    " "
## 5  ( 1 ) "*"  " "    " "     "*"       "*"     " "     " "    " "
## 6  ( 1 ) "*"  " "    " "     "*"       "*"     " "     " "    " "
## 7  ( 1 ) " "  " "    " "     "*"       "*"     " "     " "    " "
## 8  ( 1 ) " "  "*"    " "     "*"       "*"     " "     " "    " "
```

```r
fit <- regsubsets(Salary~., data=df, nvmax = 19)
fit_sum <- summary(fit)
```

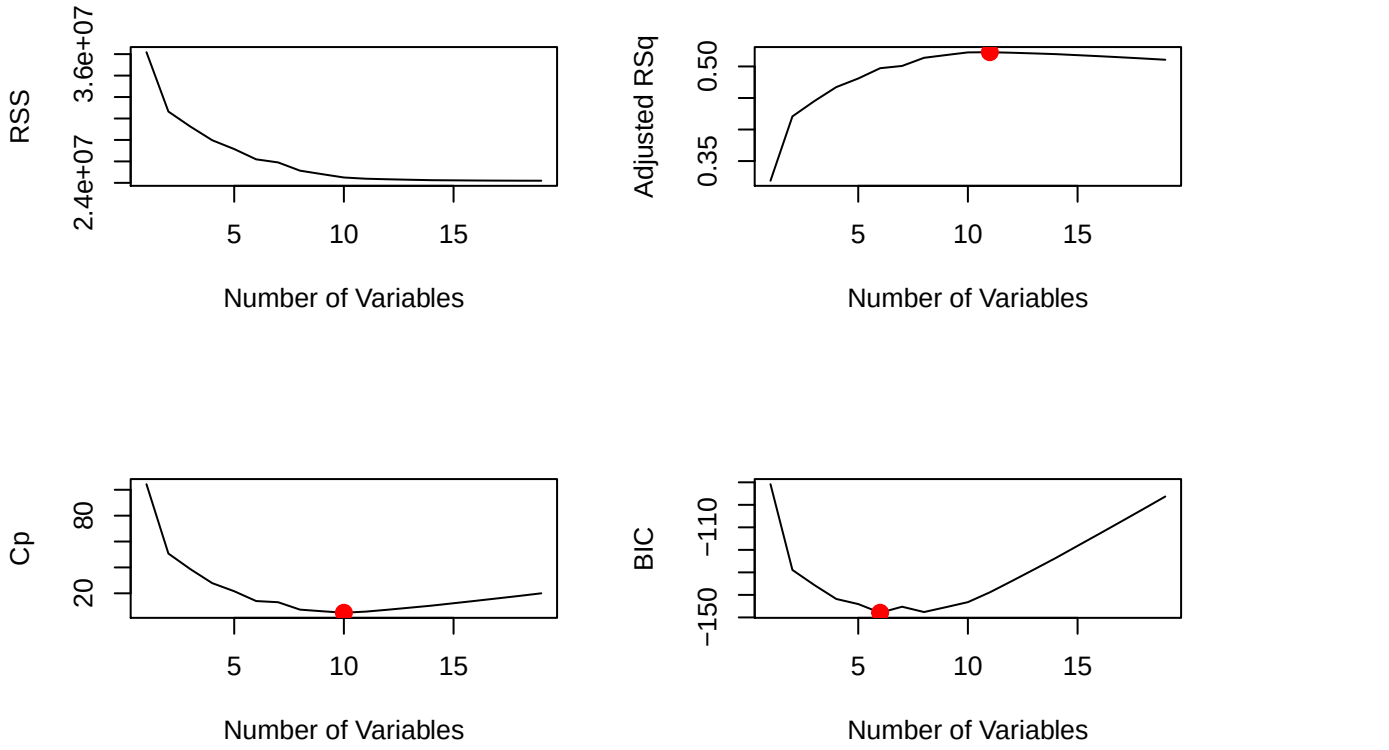
```
names(fit_sum)
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

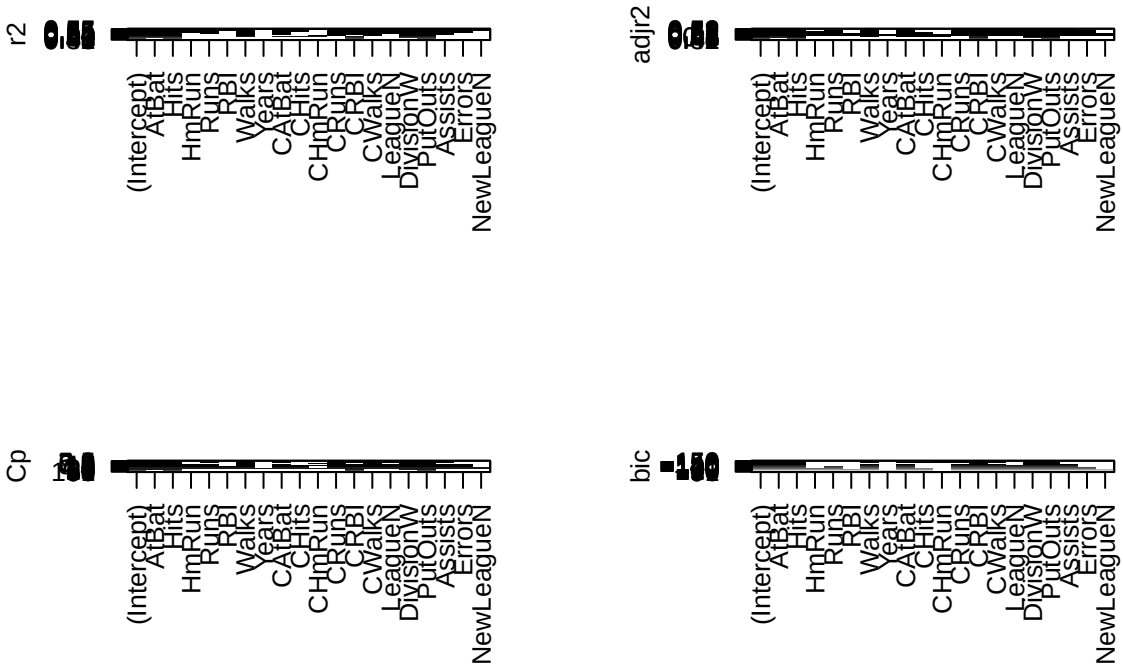
```
fit_sum$rsq
```

```
##  [1] 0.3214501 0.4252237 0.4514294 0.4754067 0.4908036 0.5087146 0.5141227
##  [8] 0.5285569 0.5346124 0.5404950 0.5426153 0.5436302 0.5444570 0.5452164
## [15] 0.5454692 0.5457656 0.5459518 0.5460945 0.5461159
```

```
par(mfrow = c(2,2))
plot(fit_sum$rss, xlab = "Number of Variables", ylab = "RSS", type = "l")
plot(fit_sum$adjr2, xlab = "Number of Variables", ylab = "Adjusted RSq", type = "l")
points(11, fit_sum$adjr2[11], col = "red", cex = 2, pch = 20)
plot(fit_sum$cp, xlab = "Number of Variables", ylab = "Cp", type = "l")
points(which.min(fit_sum$cp), fit_sum$cp[which.min(fit_sum$cp)], col = "red", cex = 2, pch = 20)
plot(fit_sum$bic, xlab = "Number of Variables", ylab = "BIC", type = "l")
points(which.min(fit_sum$bic), fit_sum$bic[which.min(fit_sum$bic)], col = "red", cex = 2, pch = 20)
```




```
plot(fit, scale = "r2")
plot(fit, scale = "adjr2")
plot(fit, scale = "Cp")
plot(fit, scale = "bic")
```




```
coef(fit, 7)
```

```
##  (Intercept)         Hits        Walks       CAtBat        CHits
##   79.4509472    1.2833513    3.2274264   -0.3752350    1.4957073
##       CHmRun    DivisionW      PutOuts
##    1.4420538 -129.9866432    0.2366813
```

```
#Ridge Regression--------
x <- model.matrix(Salary~., df)[,-1]
y <- df$Salary

# A taste of it
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following object is masked from 'package:tidyr':
##
##     expand
```

```
## Loading required package: foreach
```

```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
## Loaded glmnet 2.0-13
```

```
grid <- 10^seq(10, -2, length = 100)
ridge.fit <- glmnet(x, y , alpha = 0 , lambda = grid)
```

```r
#100 columes 20 values of coefficient including intercept
dim(coef(ridge.fit))
```

```
## [1]  20 100
```

```r
#value of coefficient when lambda =
ridge.fit$lambda[50]
```

```
## [1] 11497.57
```

```r
coef(ridge.fit)[,50]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
## 407.356050200   0.036957182   0.138180344   0.524629976   0.230701523
##           RBI         Walks         Years        CAtBat         CHits
##   0.239841459   0.289618741   1.107702929   0.003131815   0.011653637
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
##   0.087545670   0.023379882   0.024138320   0.025015421   0.085028114
##     DivisionW       PutOuts       Assists        Errors    NewLeagueN
##  -6.215440973   0.016482577   0.002612988  -0.020502690   0.301433531
```

```r
#Training and Test
set.seed(1)
train <- sample(1:nrow(x), nrow(x)/2)
test <- (-train)
y.test <- y[test]


#lambda = 4
ridge.fit <- glmnet(x[train, ], y[train], alpha = 0, lambda = grid, thresh = 1e-12)
ridge.pre <- predict(ridge.fit, s = 4, newx = x[test, ])
mean((ridge.pre - y.test)^2)
```

```
## [1] 101036.8
```

```r
#using only average of y to predit
mean((mean(y[train]) - y.test)^2)
```

```
## [1] 193253.1
```

```r
#lambda very large
ridge.pre <- predict(ridge.fit, s = 1e10, newx = x[test, ])
mean((ridge.pre - y.test)^2)
```

```
## [1] 193253.1
```

```r
#OLS is just ridge regression with lambda = 0

#10fold cross validation
set.seed(1)
cv.out <- cv.glmnet(x[train,], y[train], alpha = 0)
plot(cv.out)
bestlambda <- cv.out$lambda.min
log(bestlambda)
```

```
## [1] 5.355367
```

```r
ridge.pre <- predict(ridge.fit, s = bestlambda, newx = x[test, ])
mean((ridge.pre - y.test)^2)
```

```
## [1] 96015.51
```

```
#visialization
dev.off()
```

```
## null device
##           1
```

```
out <- glmnet(x, y , alpha = 0)
predict(out, type = "coefficients", s= bestlambda)[1:20,]
```

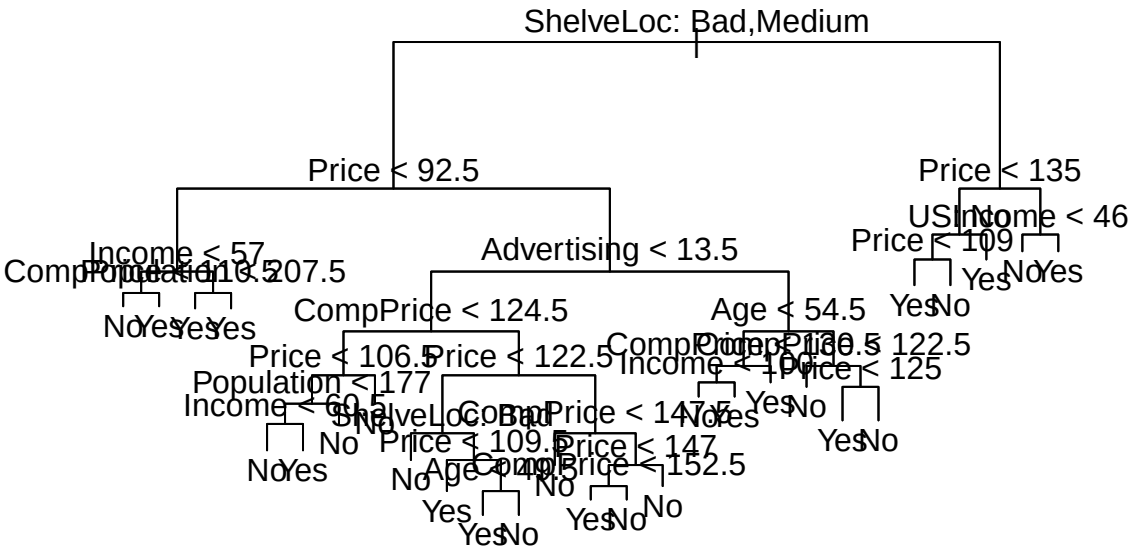
```
##  (Intercept)        AtBat         Hits        HmRun         Runs
##   9.88487157   0.03143991   1.00882875   0.13927624   1.11320781
##          RBI        Walks        Years       CAtBat        CHits
##   0.87318990   1.80410229   0.13074381   0.01113978   0.06489843
##       CHmRun        CRuns         CRBI       CWalks      LeagueN
##   0.45158546   0.12900049   0.13737712   0.02908572  27.18227535
##    DivisionW      PutOuts      Assists       Errors   NewLeagueN
## -91.63411299   0.19149252   0.04254536  -1.81244470   7.21208390
```

Tree Based Model

```
library(tree)
df <- Carseats
High <- ifelse(df$Sales <= 8, "No", "Yes")
df <- data.frame(df, High)

tree.fit <- tree(High~. - Sales, df)
summary(tree.fit)
```

```
##
## Classification tree:
## tree(formula = High ~ . - Sales, data = df)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Income"      "CompPrice"   "Population"
## [6] "Advertising" "Age"         "US"
## Number of terminal nodes:  27
## Residual mean deviance:  0.4575 = 170.7 / 373
## Misclassification error rate: 0.09 = 36 / 400
```

```
plot(tree.fit)
text(tree.fit, pretty = 0)
```




```
#prediction
set.seed(2)
train <- sample(1:nrow(df), 200)
df.test <- df[-train, ]
High.test <- High[-train]

tree.fit <- tree(High~.-Sales, df, subset = train)
tree.pred <- predict(tree.fit, df.test, type = "class")
table(tree.pred, High.test)
```

```
##          High.test
## tree.pred No Yes
##       No  86  27
##       Yes 30  57
```

```
(86+57)/200
```

```
## [1] 0.715
```

```
#random forest
library(MASS)
```

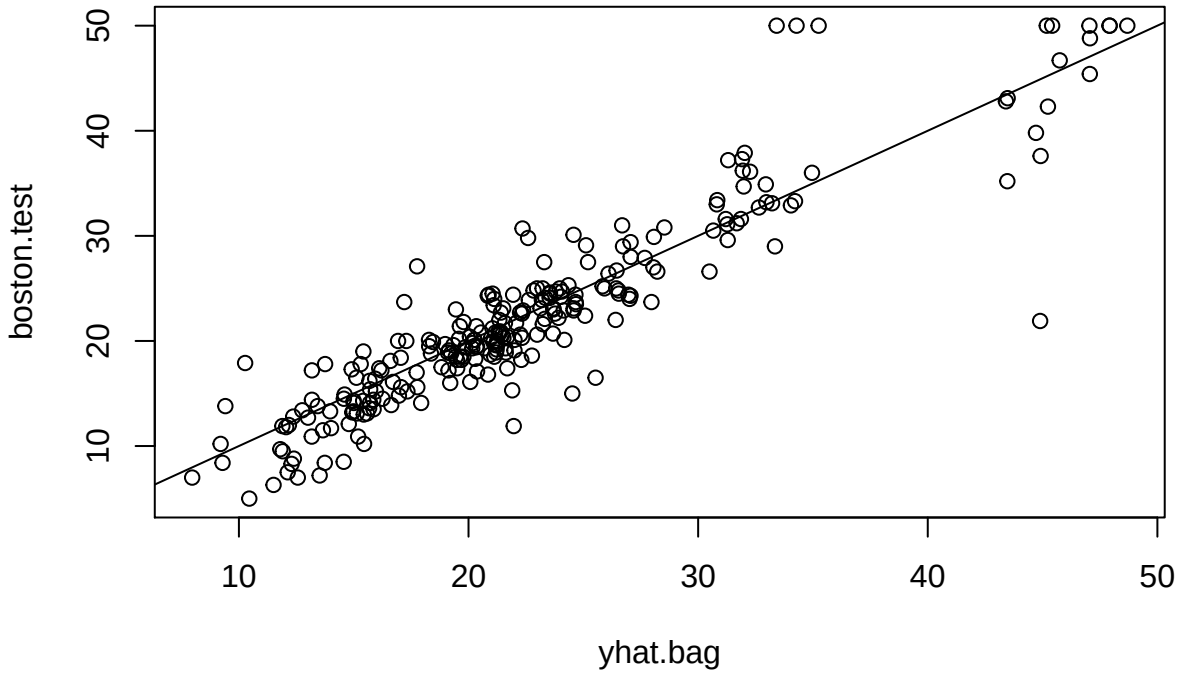
```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(randomForest)
```

```
## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(1)
train <- sample(1:nrow(Boston), nrow(Boston)/2)
boston.test <- Boston[-train, "medv"]

#bagging
bag.boston <- randomForest(medv~., data = Boston, subset = train, mtry = 13, importance = T)
yhat.bag <- predict(bag.boston, newdata = Boston[-train,])
plot(yhat.bag, boston.test)
abline(0,1)
```




```
mean((yhat.bag - boston.test)^2)
```

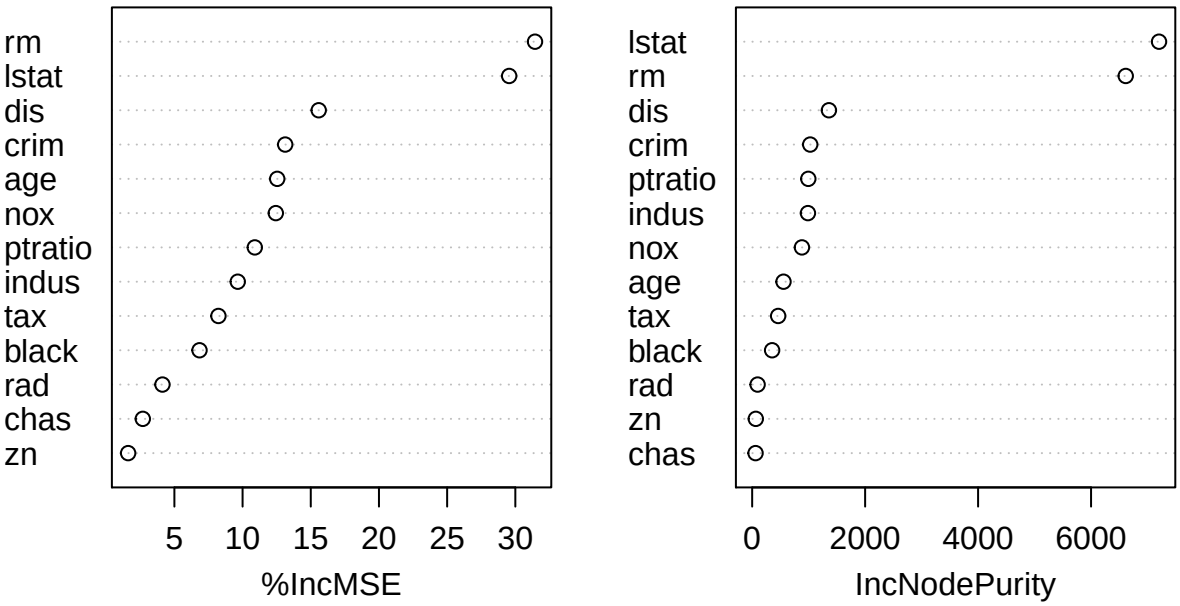
```
## [1] 13.33831
```

```
#random forest
rf.boston <- randomForest(medv~., data = Boston, subset = train, mtry = 6, importance = T)
yhat.rf <- predict(rf.boston, newdata = Boston[-train,])
mean((yhat.rf - boston.test)^2)
```

```
## [1] 11.36948
```

```
#variable importance
#1 the mean decrease o faccuracy in predictions on the out of bag samples when a given variables is exc
#2.
varImpPlot(rf.boston)
```

rf.boston


rm
lstat
dis
crim
age
nox
ptratio
indus
tax
black
rad
chas
zn
5
10
15
20
25
30
%IncMSE
lstat
rm
dis
crim
ptratio
indus
nox
age
tax
black
rad
zn
chas
0
2000
4000
6000
IncNodePurity