# R을 활용한 의생명 데이터 분석 - Part 1. R의 Basic and Advanced Analysis

*Wednesday May 29 2019*

## Contents

# 1 Assignment

```
x = 2
x
```

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

```
x <- 3
X
```

```
## Error in eval(expr, envir, enclos): object 'X' not found
```

```
ls()
```

```
## [1] "exer.num" "lab.num"  "x"
```

```
rm(x)
x
```

```
## Error in eval(expr, envir, enclos): object 'x' not found
```

```
ls()
```

```
## [1] "exer.num" "lab.num"
```

# 2 Arithmetic operators

**사칙 연산의 우선순위** ^ = ** > * = / > + = -

```r
7 * 3
```

```
## [1] 21
```

```r
7 + 2 * 3
```

```
## [1] 13
```

```r
(7 + 2) * 3
```

```
## [1] 27
```

```r
12/2 + 4
```

```
## [1] 10
```

```r
12/(2 + 4)
```

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

```r
3^2
```

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

```r
3^2
```

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

```r
2 * 3^2
```

```
## [1] 18
```

# 3 Vector

## 3.1 Combine

```r
x = c(1.5, 2, 2.5)
x
```

```
## [1] 1.5 2.0 2.5
```

```r
x^2
```

```
## [1] 2.25 4.00 6.25
```

```r
x = c(x, 3)
x
```

```
## [1] 1.5 2.0 2.5 3.0
```

```r
y = c("This", "is", "an", "example")
y
```

```
## [1] "This"    "is"      "an"      "example"
```

```r
z = c(x, "x")
z
```

```
## [1] "1.5" "2"   "2.5" "3"   "x"
```

### 3.2 Sequence

```r
seq(1, 10, 1)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
seq(123, 132, 1)
```

```
##  [1] 123 124 125 126 127 128 129 130 131 132
```

```r
seq(0, 1, 0.1)
```

```
##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

```r
seq(1, 10)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
seq(123, 132)
```

```
##  [1] 123 124 125 126 127 128 129 130 131 132
```

```r
a = seq(10)
a
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
b = 1:10
b
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
0:10/10
```

```
##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

```r
seq(5, 1, -1)
```

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

```r
5:1
```

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

### 3.3 Use of brakcets

```r
x = seq(0, 20, 10)
x
```

```
## [1]  0 10 20
```

```r
x[1]
```

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

```r
x[2]
```

```
## [1] 10
```

```r
x[3]
```

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

```r
x[4]
```

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

```r
x[1:2]
```

```
## [1]  0 10
```

```r
x[c(1, 3)]
```

```
## [1]  0 20
```

```r
x[-1]
```

```
## [1] 10 20
```

```r
x[-c(1:2)]
```

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

```r
y = 1:2
x[-y]
```

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

```r
x = seq(0, 20, 5)
x
```

```
## [1]  0  5 10 15 20
```

```r
x[1]
```

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

```r
x[2]
```

```
## [1] 5
```

```r
x[5]
```

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

```r
x[7]
```

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

```r
x[1:4]
```

```
## [1]  0  5 10 15
```

```r
x[c(1, 5, 3)]
```

```
## [1]  0 20 10
```

```r
x[-1]
```

```
## [1]  5 10 15 20
```

```r
x[-c(1:2)]
```

```
## [1] 10 15 20
```

```r
y = 1:2
x[-y]
```

```
## [1] 10 15 20
```

### 3.4 Arithmetic operators

```r
a = 5 * (0:3)
b = 1:4
a + b
```

```
## [1]  1  7 13 19
```

```r
a - b
```

```
## [1] -1  3  7 11
```

```r
a * b
```

```
## [1]  0 10 30 60
```

```r
a/b
```

```
## [1] 0.000000 2.500000 3.333333 3.750000
```

```r
a^b
```

```
## [1]     0    25  1000 50625
```

## 4 Simple statistics

```r
aa = seq(4, 500, 7)
mean(aa)
```

```
## [1] 249
```

```r
sum(aa)/length(aa)
```

```
## [1] 249
```

```r
var(aa)
```

```
## [1] 20874
```

```r
bb = sum((aa - mean(aa))^2)/(length(aa) - 1)
sd(aa)
```

```
## [1] 144.4784
```

```r
sqrt(bb)
```

```
## [1] 144.4784
```

```r
median(aa)
```

```
## [1] 249
```

```r
summary(aa)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     4.0   126.5   249.0   249.0   371.5   494.0
```

```r
fivenum(aa)
```

```
## [1]   4.0 126.5 249.0 371.5 494.0
```

```r
aa = seq(4, 500, 7)
cumsum(aa)
```

```
##  [1]     4    15    33    58    90   129   175   228   288   355   429
## [12]   510   598   693   795   904  1020  1143  1273  1410  1554  1705
## [23]  1863  2028  2200  2379  2565  2758  2958  3165  3379  3600  3828
## [34]  4063  4305  4554  4810  5073  5343  5620  5904  6195  6493  6798
## [45]  7110  7429  7755  8088  8428  8775  9129  9490  9858 10233 10615
## [56] 11004 11400 11803 12213 12630 13054 13485 13923 14368 14820 15279
## [67] 15745 16218 16698 17185 17679
```

```r
rev(aa)
```

```
##  [1] 494 487 480 473 466 459 452 445 438 431 424 417 410 403 396 389 382
## [18] 375 368 361 354 347 340 333 326 319 312 305 298 291 284 277 270 263
## [35] 256 249 242 235 228 221 214 207 200 193 186 179 172 165 158 151 144
## [52] 137 130 123 116 109 102  95  88  81  74  67  60  53  46  39  32  25
## [69]  18  11   4
```

```r
min(aa)
```

```
## [1] 4
```

```r
max(aa)
```

```
## [1] 494
```

```r
quantile(aa)
```

```
##    0%   25%   50%   75%  100% 
##   4.0 126.5 249.0 371.5 494.0
```

```r
range(aa)
```

```
## [1]   4 494
```

```r
quantile(aa, c(0, 1))
```

```
##   0% 100% 
##    4  494
```

```r
quantile(aa, seq(0, 1, 0.1))
```

```
##   0%  10%  20%  30%  40%  50%  60%  70%  80%  90% 100% 
##    4   53  102  151  200  249  298  347  396  445  494
```

## 5 Logical opeators

```r
3 == 4
```

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

```r
3 < 4
```

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

```r
3 != 4
```

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

```r
x = -3:3
x
```

```
## [1] -3 -2 -1  0  1  2  3
```

```r
x < 2
```

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

```r
sum(x < 2)
```

```
## [1] 5
```

```r
sum(c(TRUE, TRUE, FALSE, TRUE))  # TRUE => 1, FALSE => 0
```

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

```r
y = 1:30
y%%2
```

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

```r
z = y%%2
z == 0
```

```
##  [1] FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE
## [12]  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE
## [23] FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE
```

```r
sum(z == 0)
```

```
## [1] 15
```

```r
sum(z == 0) == 0
```

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

```r
sum(z[seq(1, length(z), 2)]%%2) == 0
```

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

```r
sum(z[seq(2, length(z), 2)]%%2) == 0
```

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

```r
A = c("A", "B", "A", "D", "B")
A == "A"
```

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

```r
A == "B"
```

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

```r
A[A == "A"]
```

```
## [1] "A" "A"
```

```r
A[A == "B"]
```

```
## [1] "B" "B"
```

```r
A[A != "A"]
```

```
## [1] "B" "D" "B"
```

```r
A[A != "B"]
```

```
## [1] "A" "A" "D"
```

```r
A[A != "A" & A != "B"]
```

```
## [1] "D"
```

```r
A[A != "A" | A != "B"]
```

```
## [1] "A" "B" "A" "D" "B"
```

```r
which(A == "A")
```

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

```r
A[which(A == "A")]
```

```
## [1] "A" "A"
```

```r
A[-which(A == "A")]
```

```
## [1] "B" "D" "B"
```

```r
weight = 60:68
height = c(seq(120, 155, 5), 135)
weight
```

```
## [1] 60 61 62 63 64 65 66 67 68
```

```r
height
```

```
## [1] 120 125 130 135 140 145 150 155 135
```

```r
height < 140
```

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

```r
height[height < 140]
```

```
## [1] 120 125 130 135 135
```

```r
weight[weight > 65]
```

```
## [1] 66 67 68
```

```r
height[height < 140 & height != 120]
```

```
## [1] 125 130 135 135
```

## 5.1 실습문제 1

몸무게가 65보다 작은 사람들의 몸무게 평균을 구하시오

```
## [1] 62
```

## 5.2 실습문제 2

몸무게가 65보다 큰 사람들의 키 평균을 구하시오

```
## [1] 146.6667
```

## 5.3 실습문제 3

키가 130보다 작은 사람들의 몸무게 평균을 구하시오.

```
## [1] 60.5
```

# 6 Data structure

## 6.1 Factor

Factor는 R에서 category 변수를 효율적으로 표현하기 위해서 사용

**Character type vector를 factor로 변환하기**

```
gender <- c("Male", "Female", "Female", "Male")
str(gender)
```

```
##  chr [1:4] "Male" "Female" "Female" "Male"
```

```
factor_gender <- factor(gender)  # factor_gender has two levels called Male and Female
factor_gender
```

```
## [1] Male   Female Female Male
## Levels: Female Male
```

```
str(factor_gender)
```

```
##  Factor w/ 2 levels "Female","Male": 2 1 1 2
```

```
levels(factor_gender)
```

```
## [1] "Female" "Male"
```

**factor를 원래 type으로 변환하기**

```
levels(factor_gender)[factor_gender]
```

```
## [1] "Male"   "Female" "Female" "Male"
```

## 6.2 Matrix

### 6.2.1 Matrix 만들기

```
m <- matrix(1:6)
matrix(1:6, nrow = 2)
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```r
matrix(1:6, nrow = 2, byrow = T)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

```r
x = 3:8
matrix(x, 3, 2)
```

```
##      [,1] [,2]
## [1,]    3    6
## [2,]    4    7
## [3,]    5    8
```

```r
matrix(x, ncol = 2)
```

```
##      [,1] [,2]
## [1,]    3    6
## [2,]    4    7
## [3,]    5    8
```

```r
matrix(x, ncol = 3)
```

```
##      [,1] [,2] [,3]
## [1,]    3    5    7
## [2,]    4    6    8
```

```r
matrix(x, ncol = 3, byrow = T)
```

```
##      [,1] [,2] [,3]
## [1,]    3    4    5
## [2,]    6    7    8
```

### 6.2.2 Matrix 연산

```r
A = matrix(0:5, 2, 3)
B = matrix(seq(0, 10, 2), 2, 3)
A + B
```

```
##      [,1] [,2] [,3]
## [1,]    0    6   12
## [2,]    3    9   15
```

```r
A - B
```

```
##      [,1] [,2] [,3]
## [1,]    0   -2   -4
## [2,]   -1   -3   -5
```

```r
A * B
```

```
##      [,1] [,2] [,3]
## [1,]    0    8   32
## [2,]    2   18   50
```

```r
t(A) %*% B
```

```
##      [,1] [,2] [,3]
## [1,]    2    6   10
```

```
## [2,]    6   26   46
## [3,]   10   46   82
```

```r
A %*% t(B)
```

```
##      [,1] [,2]
## [1,]   40   52
## [2,]   52   70
```

### 6.2.3 Matrix 값 얻기/변경하기

```r
data = c(197, 8, 1.8, 1355, 58, 1.7, 2075, 81, 1.8)
country.data = matrix(data, nrow = 3, byrow = T)
country.data
```

```
##      [,1] [,2] [,3]
## [1,]  197    8  1.8
## [2,] 1355   58  1.7
## [3,] 2075   81  1.8
```

```r
dim(country.data)
```

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

```r
country.data[1, 2]
```

```
## [1] 8
```

```r
country.data[2, 1:3]
```

```
## [1] 1355.0   58.0    1.7
```

```r
country.data[2, ]
```

```
## [1] 1355.0   58.0    1.7
```

```r
country.data[1, 2] = 10
country.data
```

```
##      [,1] [,2] [,3]
## [1,]  197   10  1.8
## [2,] 1355   58  1.7
## [3,] 2075   81  1.8
```

```r
country.data[1, 2] = 8
country.data
```

```
##      [,1] [,2] [,3]
## [1,]  197    8  1.8
## [2,] 1355   58  1.7
## [3,] 2075   81  1.8
```

### 6.2.4 Dimension names

```r
dimnames(country.data)
```

```
## NULL
```

```r
countries = c("Austria", "France", "Germany")
variables = c("GDP", "Pop", "Inflation")
dimnames(country.data) = list(countries, variables)
country.data
```

```
##          GDP Pop Inflation
## Austria  197   8       1.8
## France  1355  58       1.7
## Germany 2075  81       1.8
```

```r
dimnames(country.data)
```

```
## [[1]]
## [1] "Austria" "France"  "Germany"
##
## [[2]]
## [1] "GDP"       "Pop"       "Inflation"
```

```r
country.data["Austria", "Pop"]
```

```
## [1] 8
```

```r
country.data["France", "Inflation"] <- 2.5
country.data["France", "Inflation"]
```

```
## [1] 2.5
```

```r
country.data
```

```
##          GDP Pop Inflation
## Austria  197   8       1.8
## France  1355  58       2.5
## Germany 2075  81       1.8
```

### 6.2.5 Matrix 합치기

```r
Area = c(84, 544, 358)
country.data = cbind(country.data, Area)
country.data
```

```
##          GDP Pop Inflation Area
## Austria  197   8       1.8   84
## France  1355  58       2.5  544
## Germany 2075  81       1.8  358
```

```r
Switzerland = c(256, 7, 1.8, 41)
country.data = rbind(country.data, Switzerland)
country.data
```

```
##              GDP Pop Inflation Area
## Austria      197   8       1.8   84
## France      1355  58       2.5  544
## Germany     2075  81       1.8  358
## Switzerland  256   7       1.8   41
```

## 6.3 Data frame

```
name <- c("joe", "jhon", "Nancy")
sex <- c("M", "M", "F")
age <- c(27, 26, 26)
foo <- data.frame(name, sex, age)
foo
```

```
##    name sex age
## 1   joe   M  27
## 2  jhon   M  26
## 3 Nancy   F  26
```

```
rownames(foo)
```

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

```
colnames(foo)
```

```
## [1] "name" "sex"  "age"
```

```
country.data1
```

```
## Error in eval(expr, envir, enclos): object 'country.data1' not found
```

```
EU = c("EU", "EU", "EU", "non-EU")
country.data1 = cbind(country.data, EU)
country.data1
```

```
##             GDP    Pop  Inflation Area  EU
## Austria     "197"  "8"  "1.8"     "84"  "EU"
## France      "1355" "58" "2.5"     "544" "EU"
## Germany     "2075" "81" "1.8"     "358" "EU"
## Switzerland "256"  "7"  "1.8"     "41"  "non-EU"
```

```
country.frame = data.frame(country.data, EU, stringsAsFactors = FALSE)
country.frame
```

```
##              GDP Pop Inflation Area     EU
## Austria      197   8       1.8   84     EU
## France      1355  58       2.5  544     EU
## Germany     2075  81       1.8  358     EU
## Switzerland  256   7       1.8   41 non-EU
```

```
str(country.frame)
```

```
## 'data.frame':    4 obs. of  5 variables:
##  $ GDP      : num  197 1355 2075 256
##  $ Pop      : num  8 58 81 7
##  $ Inflation: num  1.8 2.5 1.8 1.8
##  $ Area     : num  84 544 358 41
##  $ EU       : chr  "EU" "EU" "EU" "non-EU"
```

```
country.frame["Austria", "Pop"]
```

```
## [1] 8
```

```
country.frame[, "Pop"]
```

```
## [1]  8 58 81  7
```

```r
country.frame$Pop
```

```
## [1]  8 58 81  7
```

```r
summary(country.frame)
```

```
##       GDP              Pop          Inflation          Area
##  Min.   : 197.0   Min.   : 7.00   Min.   :1.800   Min.   : 41.00
##  1st Qu.: 241.2   1st Qu.: 7.75   1st Qu.:1.800   1st Qu.: 73.25
##  Median : 805.5   Median :33.00   Median :1.800   Median :221.00
##  Mean   : 970.8   Mean   :38.50   Mean   :1.975   Mean   :256.75
##  3rd Qu.:1535.0   3rd Qu.:63.75   3rd Qu.:1.975   3rd Qu.:404.50
##  Max.   :2075.0   Max.   :81.00   Max.   :2.500   Max.   :544.00
##       EU
##  Length:4
##  Class :character
##  Mode  :character
##
##
##
```

## 6.4 Subsetting

```r
country.data[country.data[, "GDP"] > 1000, ]
```

```
##          GDP Pop Inflation Area
## France  1355  58       2.5  544
## Germany 2075  81       1.8  358
```

```r
country.data[country.data[, "GDP"] > 1000, "Area"]
```

```
##  France Germany
##     544     358
```

```r
country.data[country.data[, "GDP"] > 1000, "Area", drop = FALSE]
```

```
##         Area
## France   544
## Germany  358
```

```r
country.frame[country.frame[, "GDP"] > 1000, ]
```

```
##          GDP Pop Inflation Area EU
## France  1355  58       2.5  544 EU
## Germany 2075  81       1.8  358 EU
```

```r
country.frame[country.frame[, "GDP"] > 1000, "Area"]
```

```
## [1] 544 358
```

```r
country.frame[country.frame[, "GDP"] > 1000, "Area", drop = FALSE]
```

```
##         Area
## France   544
## Germany  358
```

```r
country.frame[country.frame$GDP > 1000, ]
```

```
##         GDP Pop Inflation Area EU
## France  1355  58       2.5  544 EU
## Germany 2075  81       1.8  358 EU
```

```r
country.frame[country.frame$GDP > 1000, "Area"]
```

```
## [1] 544 358
```

```r
country.frame[country.frame$GDP > 1000, "Area", drop = FALSE]
```

```
##         Area
## France   544
## Germany  358
```

### 6.5 실습문제 4

country.frame에서 Pop이 50이상인 나라들만으로 구성된 새로운 data.frame을 contry.frame.pop_more_than_50 이라는 이름으로 만드시오.

```
##         GDP Pop Inflation Area EU
## France  1355  58       2.5  544 EU
## Germany 2075  81       1.8  358 EU
```

### 6.6 실습문제 5

country.frame에서 Pop이 50이상인 나라들의 평균 Area는?

```
## [1] 451
```

## 7 For loop

```r
for (i in 1:4) {
    print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
```

```r
for (i in 1:4) {
    max.col = max(country.data[, i])
    print(max.col)
}
```

```
## [1] 2075
## [1] 81
## [1] 2.5
## [1] 544
```

```r
for (i in 1:4) {
    sum.col = sum(country.data[, i])
    print(sum.col)
}
```

```
## [1] 3883
## [1] 154
## [1] 7.9
## [1] 1027
```

```r
for (variable.name in colnames(country.data)) {
    print(variable.name)
}
```

```
## [1] "GDP"
## [1] "Pop"
## [1] "Inflation"
## [1] "Area"
```

```r
for (variable.name in colnames(country.data)) {
    sum.col = sum(country.data[, variable.name])
    print(sum.col)
}
```

```
## [1] 3883
## [1] 154
## [1] 7.9
## [1] 1027
```

# 8 apply function

```r
apply(country.data, 2, max)
```

```
##       GDP       Pop Inflation      Area
##    2075.0      81.0       2.5     544.0
```

```r
country.data.colMax <- apply(country.data, 2, max)
print(country.data.colMax)
```

```
##       GDP       Pop Inflation      Area
##    2075.0      81.0       2.5     544.0
```

```r
names(country.data.colMax)
```

```
## [1] "GDP"       "Pop"       "Inflation" "Area"
```

```r
print(country.data.colMax[2])
```

```
## Pop
##  81
```

```r
print(country.data.colMax["Pop"])
```

```
## Pop
##  81
```

```r
apply(country.data, 2, summary)
```

```
##            GDP   Pop Inflation   Area
## Min.     197.0  7.00     1.800  41.00
## 1st Qu.  241.2  7.75     1.800  73.25
## Median   805.5 33.00     1.800 221.00
## Mean     970.8 38.50     1.975 256.80
## 3rd Qu. 1535.0 63.75     1.975 404.50
## Max.    2075.0 81.00     2.500 544.00
```

# 9 Row-wise and column-wise functions

```r
rowSums(country.data)
```

```
##     Austria      France     Germany Switzerland
##       290.8      1959.5      2515.8       305.8
```

```r
colSums(country.data)
```

```
##       GDP       Pop Inflation      Area
##    3883.0     154.0       7.9    1027.0
```

```r
rowMeans(country.data)
```

```
##     Austria      France     Germany Switzerland
##      72.700     489.875     628.950      76.450
```

```r
colMeans(country.data)
```

```
##       GDP       Pop Inflation      Area
##   970.750    38.500     1.975   256.750
```

# 10 Function (함수) 만들기

Variance를 계산하는 함수를 만들어보겠습니다. 아래는 variance를 계산하는 수식입니다.

$$Var(x) = \frac{1}{N-1}\sum_{1}^{N}(x_i - \bar{x})^2$$

```r
New.var = function(x) {
    mean.x = mean(x)
    SS = sum((x - mean.x)^2)
    new.var = 1/(length(x) - 1) * SS
    return(new.var)
}
set.seed(2001003)
x = rnorm(100, 1, 10)
x
```

```
##   [1]  -8.0684147  -3.9999338 -11.4126851  13.0222731   3.7378879
##   [6] -10.6393883  -2.0282501  -9.1455702   0.5837701 -13.1157115
##  [11]  -0.4715538 -12.3186873 -22.6744164  -1.2479158  12.3300015
##  [16] -11.0654679  -3.7228725   2.7848346 -12.4979083  -4.7526964
```

```
## [21]  -6.1164246   3.7193410   2.9303907   0.2932760   7.5366593
## [26]   5.5690817  -7.3022533   0.9975055 -11.6412199   8.8841694
## [31]  -8.0593228   6.0870604  10.5321278  11.4284764   4.3569963
## [36]   1.3599720  10.2833344   2.1341422  18.6022983   2.7502459
## [41]   3.7022511  -5.7063314  -9.2197239   8.0700045  -7.0991974
## [46]  16.7943097   6.3521555   1.1446524 -10.3889697  -1.2511206
## [51]  12.7571097  16.3081361  -6.6782564 -14.0623208 -12.8672971
## [56]  -2.9199823 -23.8516844  10.5726719   8.1860709 -13.7913635
## [61]  19.8496475  11.0668805   1.7540156  -6.9183008  -7.9721285
## [66]  -1.7380343   5.3654448   4.2754625  -0.3851021 -20.7490859
## [71] -11.7395273  14.4064878   0.1038768  -1.9851340   3.4913320
## [76]  -8.9964043   1.5633057   7.3225562  -0.9698104   3.6741868
## [81]  24.0041358  -3.4385262   4.5616563   3.8686320  -1.5349876
## [86]   1.0661689   2.8909207 -15.9183512  -3.4458850  -6.3205048
## [91]   4.5134157  12.0473179  -3.6760487  19.2667135   4.7183274
## [96]  15.2403040   4.8059840  12.0098982  -3.4533531  -6.5614655
```

```r
var(x)
```

```
## [1] 92.99448
```

```r
New.var(x)
```

```
## [1] 92.99448
```

아래의 함수들과 apply 함수와 결합하여 연습해 보세요.

mean, sd, var, median, etc.

# 11 I/O

```r
# Breat cancer dataset
brca.cnv <- read.delim("../data/TCGA_BRCA_CNV_processed.txt")  # Copy number variation
brca.expr <- read.delim("../data/TCGA_BRCA_Expr_processed.txt")  # Gene expression

print(identical(rownames(brca.cnv), rownames(brca.expr)))
```

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

```r
head(rownames(brca.cnv))
```

```
## [1] "TCGA.A2.A0CK.01" "TCGA.BH.A1FE.01" "TCGA.BH.A0BJ.01" "TCGA.AQ.A04J.01"
## [5] "TCGA.AR.A252.01" "TCGA.AC.A2FB.01"
```

```r
head(colnames(brca.cnv))
```

```
## [1] "ABL2_CN"   "ACVR1B_CN" "AKT1_CN"   "AKT2_CN"   "AKT3_CN"   "ALK_CN"
```

```r
head(colnames(brca.expr))
```

```
## [1] "ABL2_Expr"   "ACVR1B_Expr" "AKT1_Expr"   "AKT2_Expr"   "AKT3_Expr"
## [6] "ALK_Expr"
```

```r
brca.cnv[is.na(brca.cnv)] <- 0

mean(brca.expr$ERBB2_Expr)
```

```
## [1] 7.303838
```

### 11.1 실습문제 6

주어진 dataset에서 ERBB2의 CNV가 3보다 큰 tumor sample들의 ERBB2 expression의 평균값을 구하시오.

```
## [1] 10.76321
```

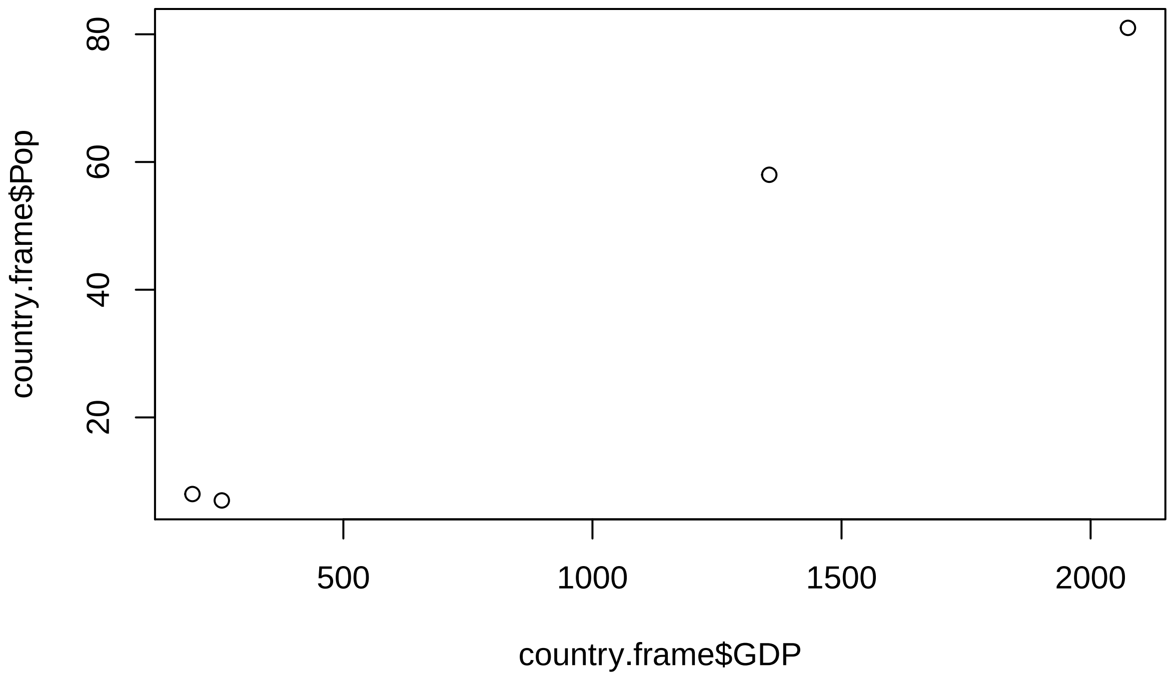
### 11.2 실습문제 7

94개 이상의 tumor sample들에서 CNV가 2 이상인 유전자를 도출하시오.

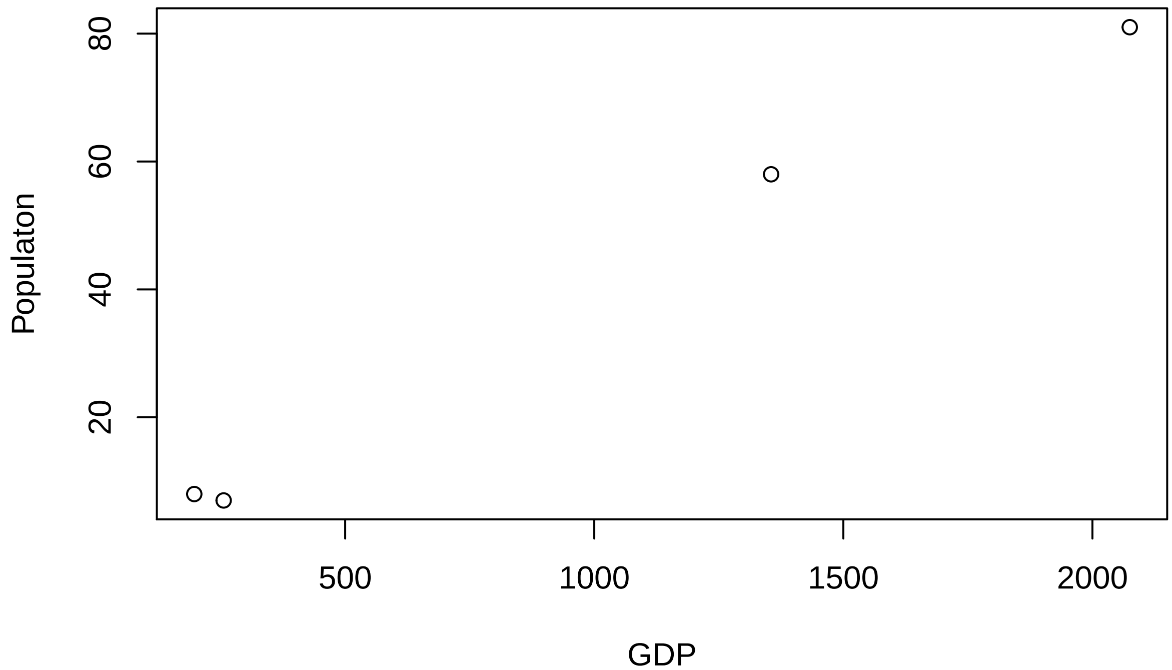
```
## [1] "CCND1_CN"   "ERBB2_CN"   "FGF19_CN"   "FGF3_CN"    "FGF4_CN"
## [6] "GPR124_CN"  "MYC_CN"     "WHSC1L1_CN" "ZNF703_CN"
```

## 12 Plot

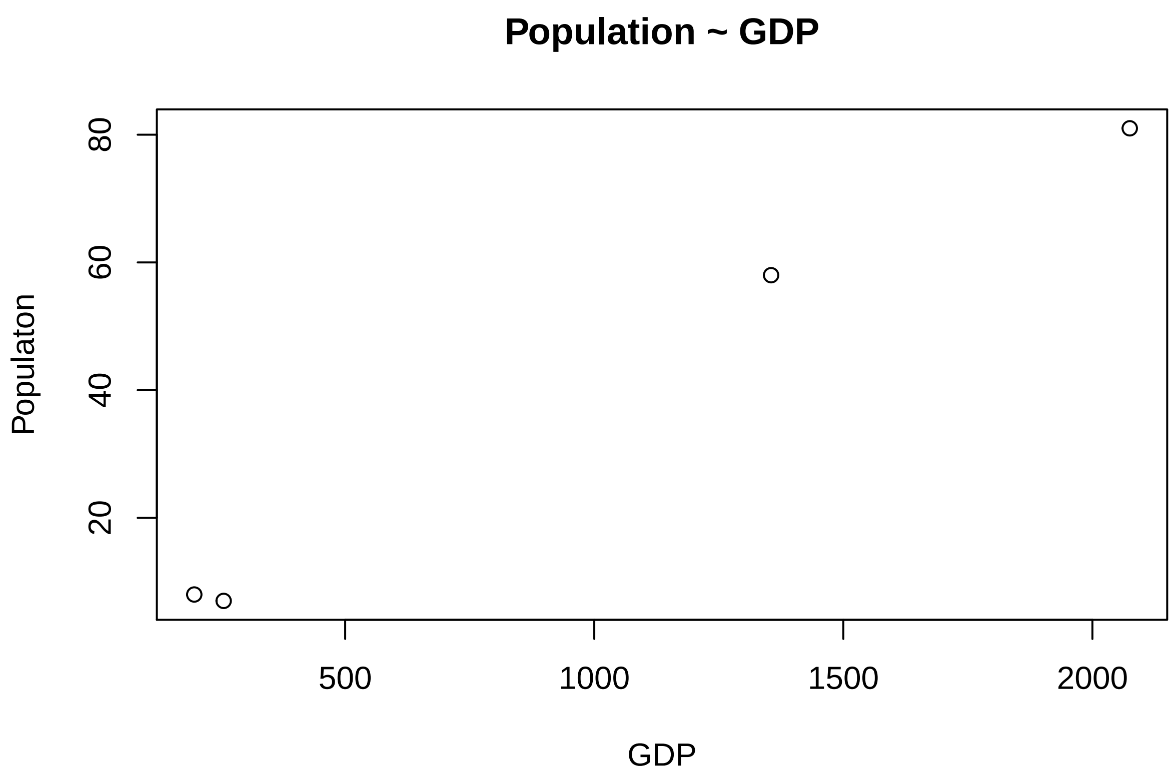
```
plot(country.frame$GDP, country.frame$Pop)
```



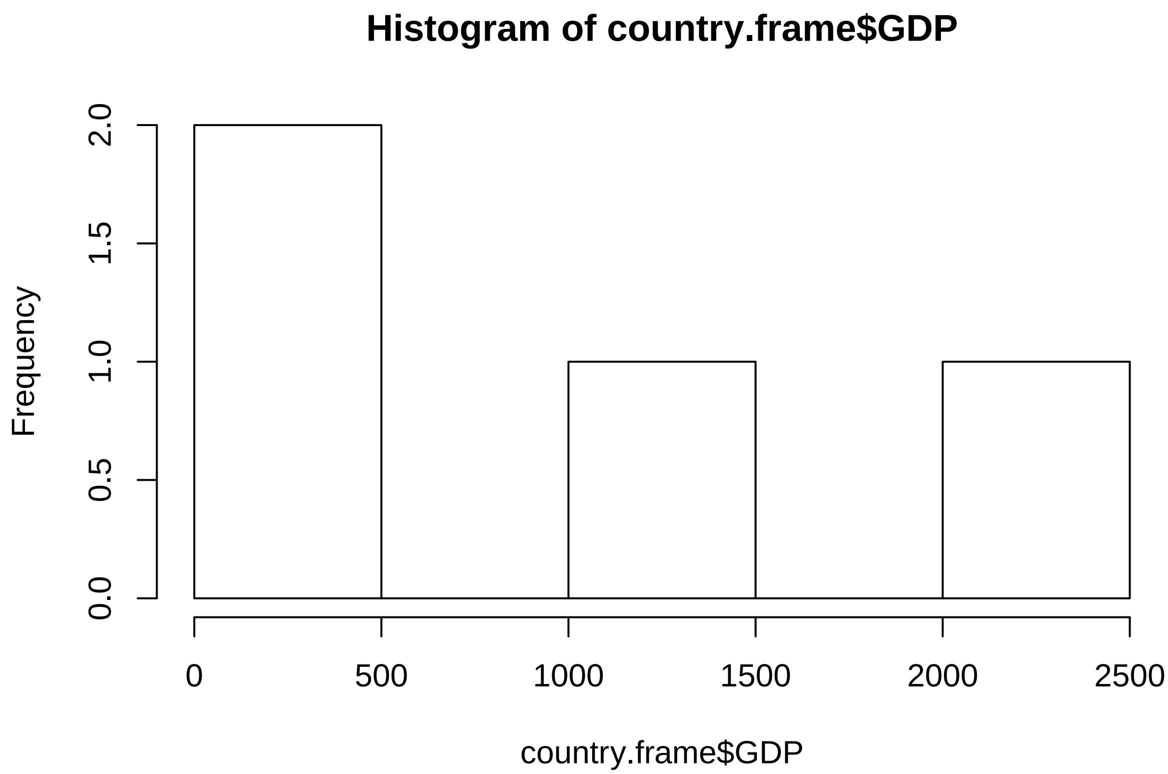

```
plot(country.frame$GDP, country.frame$Pop, xlab = "GDP", ylab = "Populaton")
```



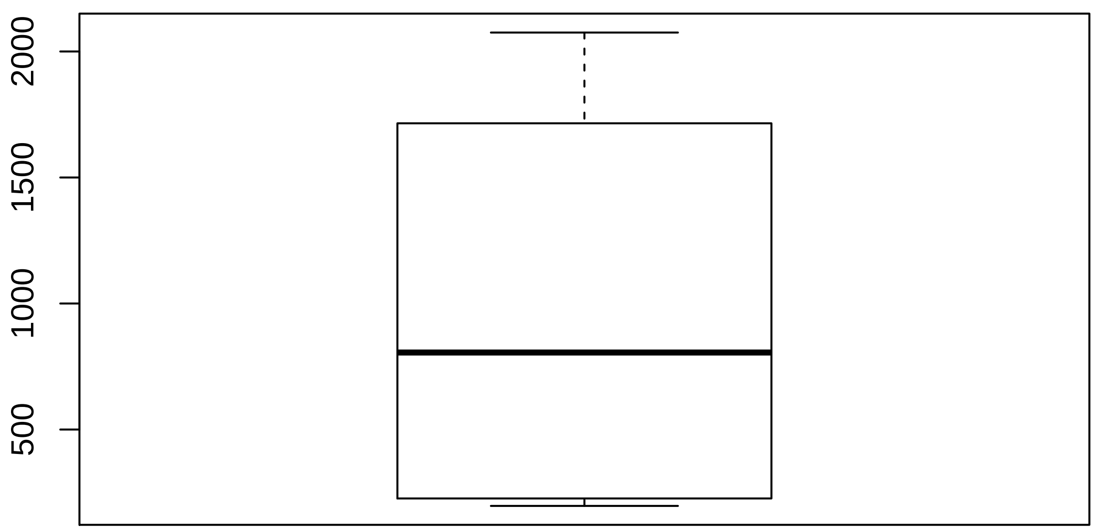

```r
plot(country.frame$GDP, country.frame$Pop, xlab = "GDP", ylab = "Populaton",
    main = "Population ~ GDP")
```




```r
hist(country.frame$GDP)
```




```
boxplot(country.frame$GDP)
```




# 13 데이터 살펴보기

### Univariate Descriptive Statistics in R

```
data(mtcars)
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 ...
```

```r
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

*mtcars*

| Variable name | Description |
|---|---:|
| [, 1] mpg | Miles/(US) gallon |
| [, 2] cyl | Number of cylinders |
| [, 3] disp | Displacement (cu.in.) |
| [, 4] hp | Gross horsepower |
| [, 5] drat | Rear axle ratio |
| [, 6] wt | Weight (1000 lbs) |
| [, 7] qsec | 1/4 mile time |
| [, 8] vs | V/S |
| [, 9] am | Transmission (0 = automatic, 1 = manual) |
| [,10] gear | Number of forward gears |
| [,11] carb | Number of carburetors |

```r
range(mtcars$mpg)
```

```
## [1] 10.4 33.9
```

```r
length(mtcars$mpg)
```

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

```r
mean(mtcars$mpg)
```

```
## [1] 20.09062
```

```r
median(mtcars$mpg)
```

```
## [1] 19.2
```

```r
sd(mtcars$mpg)
```

```
## [1] 6.026948
```

```r
var(mtcars$mpg)
```

```
## [1] 36.3241
```

```r
sd(mtcars$mpg)^2
```

```
## [1] 36.3241
```

```r
IQR(mtcars$mpg)
```

```
## [1] 7.375
```

```r
quantile(mtcars$mpg, 0.67)
```

```
##  67%
## 21.4
```

```r
max(mtcars$mpg)
```

```
## [1] 33.9
```

```r
min(mtcars$mpg)
```

```
## [1] 10.4
```

```r
cummax(mtcars$mpg)
```

```
##  [1] 21.0 21.0 22.8 22.8 22.8 22.8 22.8 24.4 24.4 24.4 24.4 24.4 24.4 24.4
## [15] 24.4 24.4 24.4 32.4 32.4 33.9 33.9 33.9 33.9 33.9 33.9 33.9 33.9 33.9
## [29] 33.9 33.9 33.9 33.9
```

```r
cummin(mtcars$mpg)
```

```
##  [1] 21.0 21.0 21.0 21.0 18.7 18.1 14.3 14.3 14.3 14.3 14.3 14.3 14.3 14.3
## [15] 10.4 10.4 10.4 10.4 10.4 10.4 10.4 10.4 10.4 10.4 10.4 10.4 10.4 10.4
## [29] 10.4 10.4 10.4 10.4
```

```r
summary(mtcars)
```

```
##       mpg             cyl             disp             hp       
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0  
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
##       drat             wt             qsec             vs        
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
##        am              gear            carb      
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000  
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
##  Median :0.0000   Median :4.000   Median :2.000  
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812  
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000  
```

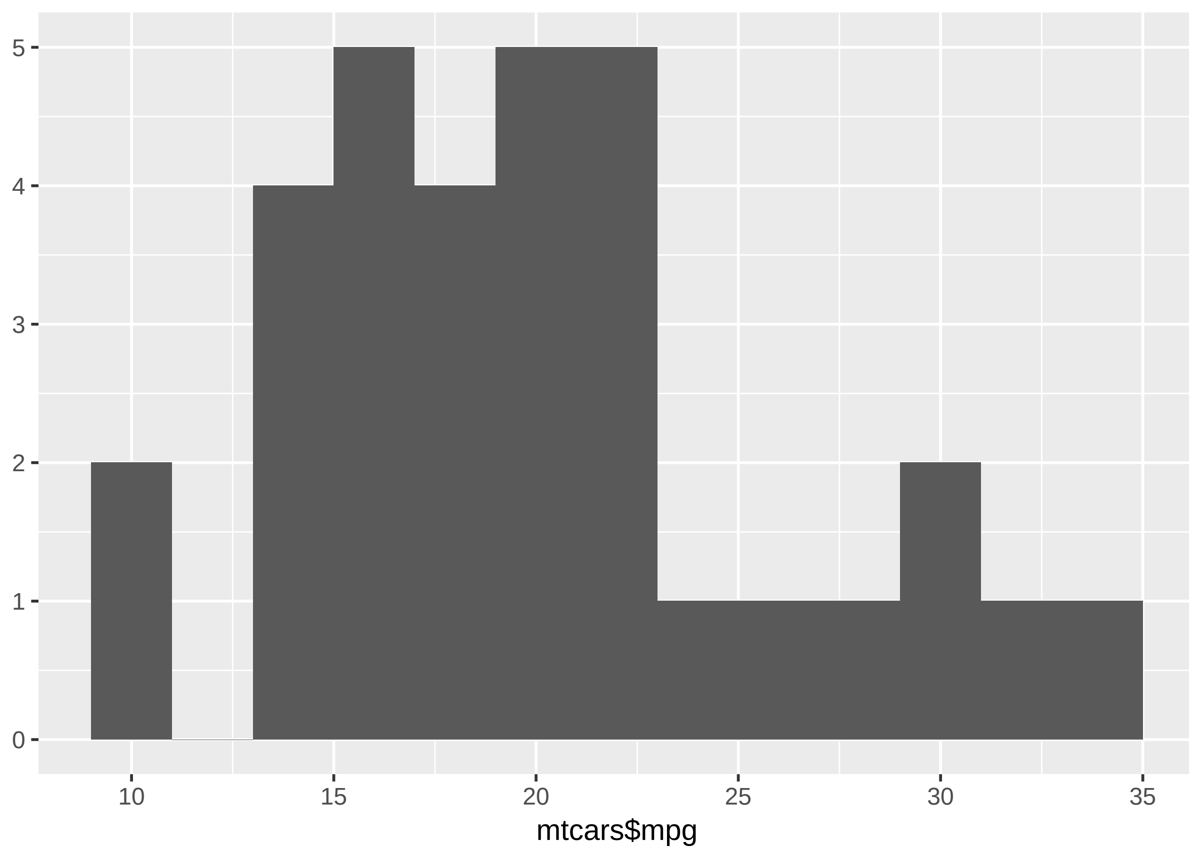
```r
table(mtcars$cyl)
```

```
##
##  4  6  8
## 11  7 14
```

```r
stem(mtcars$mpg)
```

```
##
##   The decimal point is at the |
##
##   10 | 44
##   12 | 3
##   14 | 3702258
##   16 | 438
##   18 | 17227
##   20 | 00445
##   22 | 88
##   24 | 4
##   26 | 03
##   28 |
##   30 | 44
##   32 | 49
```

```r
library(ggplot2)
qplot(mtcars$mpg, binwidth = 2)
```




```r
mode <- function(x) {
    temp <- table(x)
    names(temp)[temp == max(temp)]
```

```r
}
x = c(1, 2, 3, 3, 3, 4, 4, 5, 5, 5, 6)
mode(x)
```

```
## [1] "3" "5"
```

Correlations and Multivariate Analysis

```r
# install.packages('NMF')
library(NMF)
```

```
## Loading required package: pkgmaker
```

```
## Loading required package: registry
```

```
## Loading required package: rngtools
```

```
## Loading required package: cluster
```

```
## NMF - BioConductor layer [OK] | Shared memory capabilities [NO: bigmemory] | Cores 3/4
```

```
##   To enable shared memory capabilities, try: install.extras('
## NMF
## ')
```

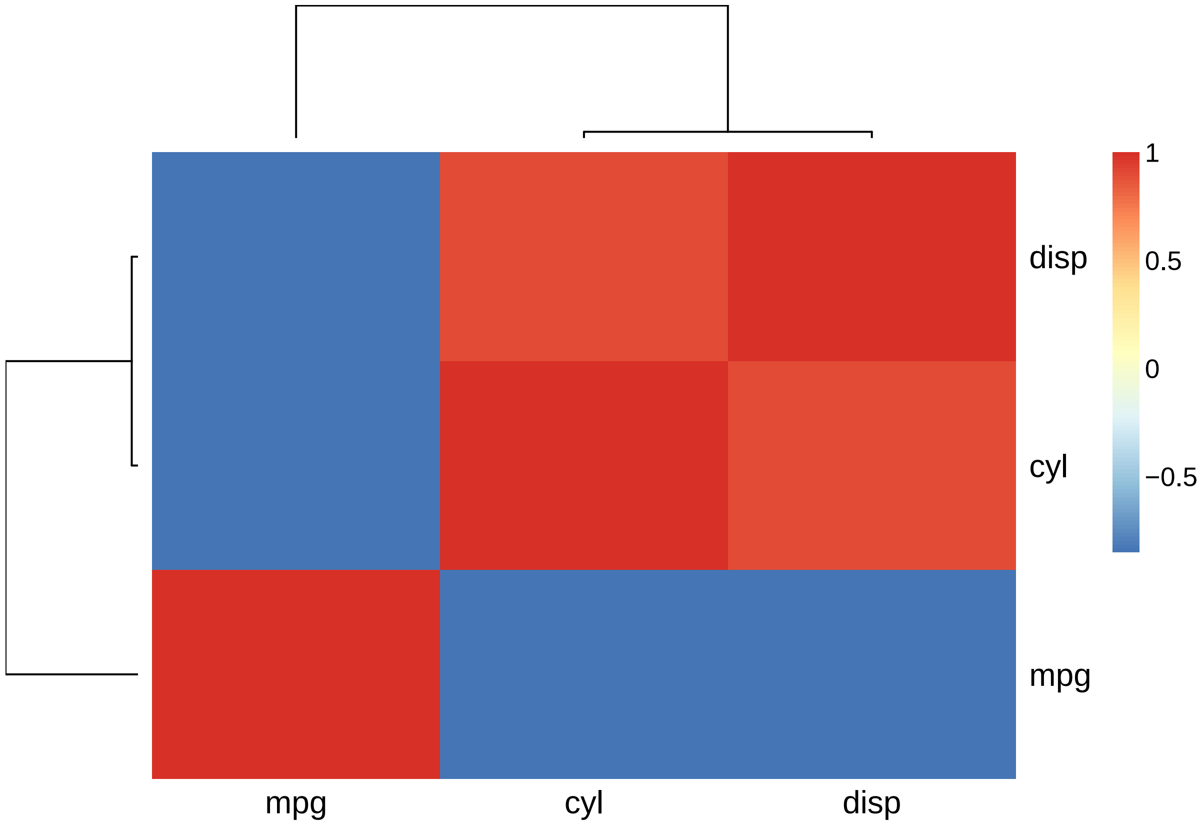
```r
cov(mtcars[1:3])
```

```
##              mpg        cyl       disp
## mpg    36.324103  -9.172379  -633.0972
## cyl    -9.172379   3.189516   199.6603
## disp -633.097208 199.660282 15360.7998
```

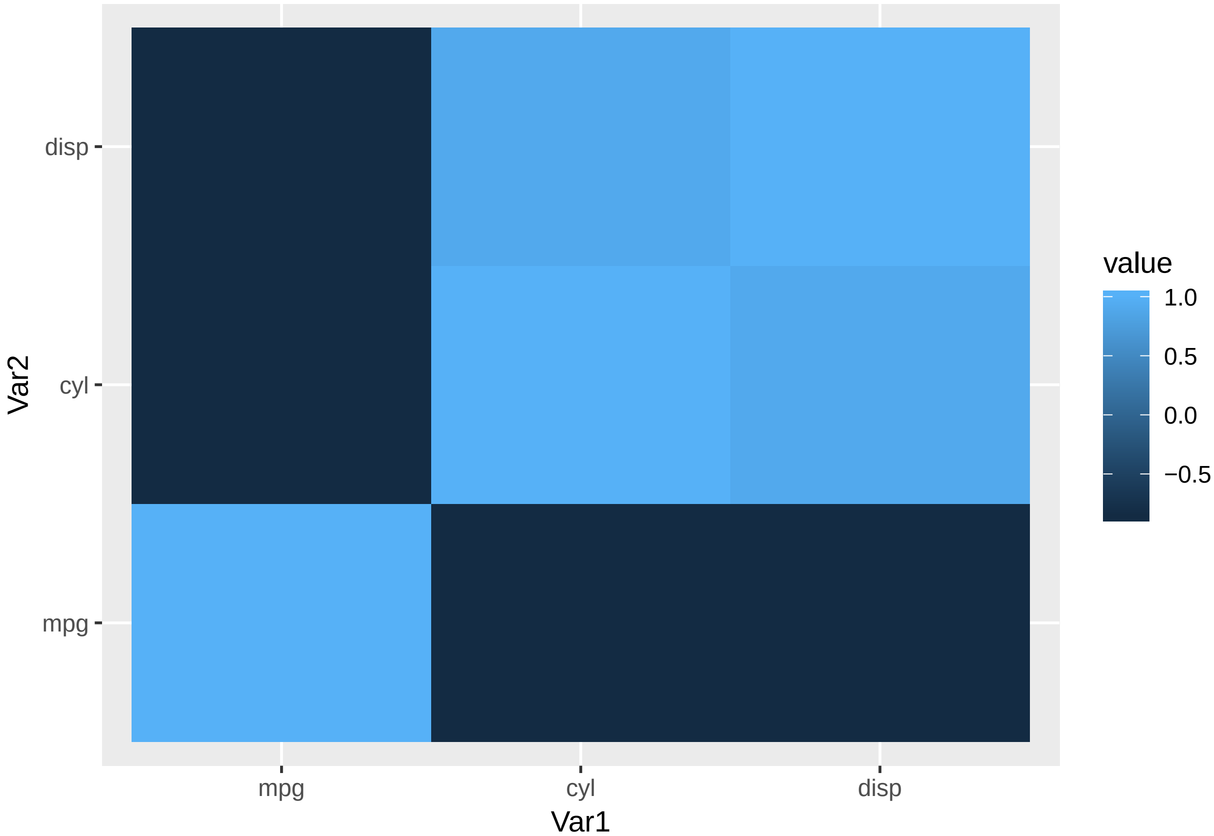
```r
cor(mtcars[1:3])
```

```
##             mpg        cyl       disp
## mpg   1.0000000 -0.8521620 -0.8475514
## cyl  -0.8521620  1.0000000  0.9020329
## disp -0.8475514  0.9020329  1.0000000
```

```r
aheatmap(cor(mtcars[1:3]))
```




```
library(reshape2)
qplot(x = Var1, y = Var2, data = melt(cor(mtcars[1:3])), fill = value, geom = "tile")
```




Linear Regression and Multivariate Analysis

```
lmfit = lm(mtcars$mpg ~ mtcars$cyl)
lmfit
```

```
##
## Call:
## lm(formula = mtcars$mpg ~ mtcars$cyl)
##
## Coefficients:
## (Intercept)   mtcars$cyl
##      37.885       -2.876
```

```
summary(lmfit)
```

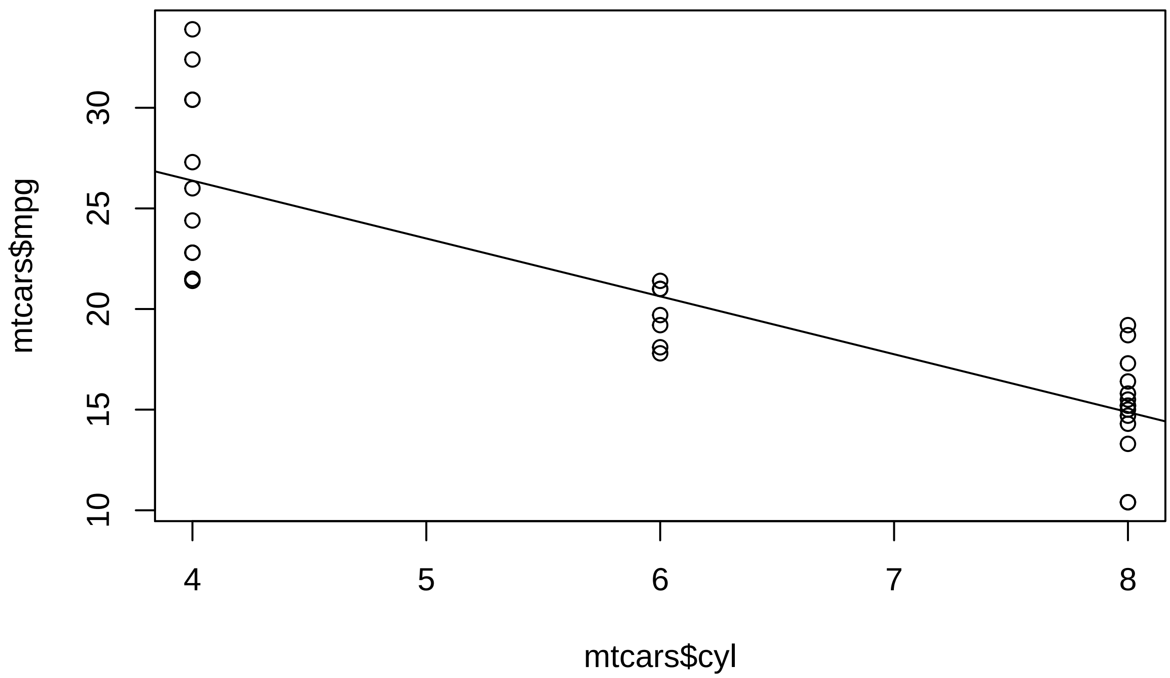
```
##
## Call:
## lm(formula = mtcars$mpg ~ mtcars$cyl)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.9814 -2.1185  0.2217  1.0717  7.5186
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.8846     2.0738   18.27  < 2e-16 ***
## mtcars$cyl   -2.8758     0.3224   -8.92 6.11e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 3.206 on 30 degrees of freedom
## Multiple R-squared:  0.7262, Adjusted R-squared:  0.7171
## F-statistic: 79.56 on 1 and 30 DF,  p-value: 6.113e-10
```

```r
anova(lmfit)
```

```
## Analysis of Variance Table
##
## Response: mtcars$mpg
##             Df Sum Sq Mean Sq F value    Pr(>F)
## mtcars$cyl   1 817.71  817.71  79.561 6.113e-10 ***
## Residuals   30 308.33   10.28
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

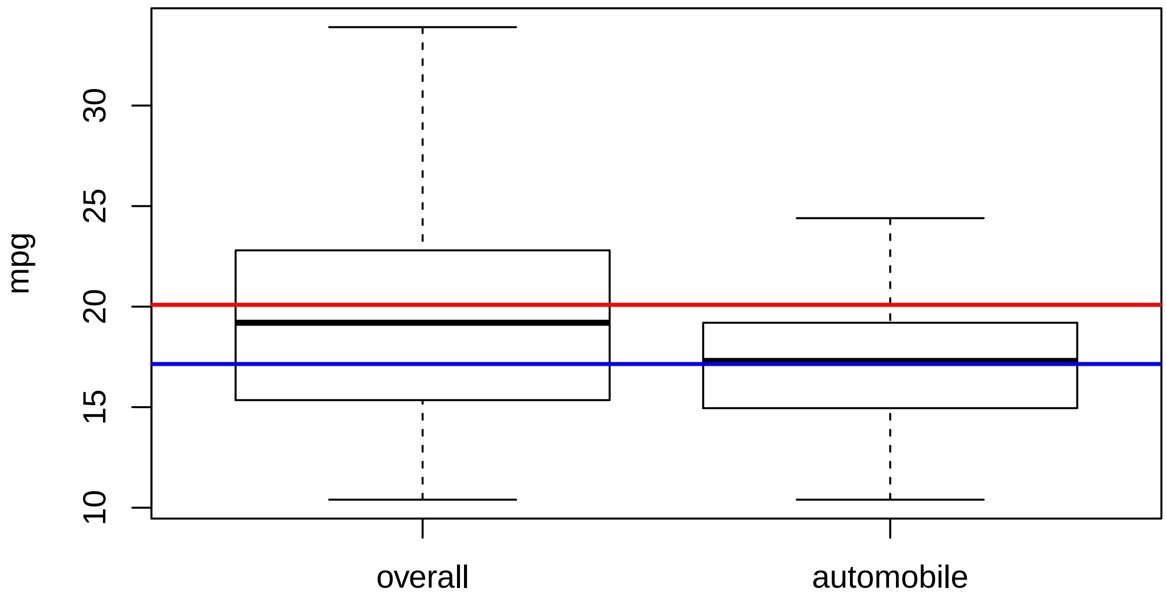
```r
lmfit = lm(mtcars$mpg ~ mtcars$cyl)
plot(mtcars$cyl, mtcars$mpg)
abline(lmfit)
```




# 14 통계 테스트

## 14.1 Student's t-Test

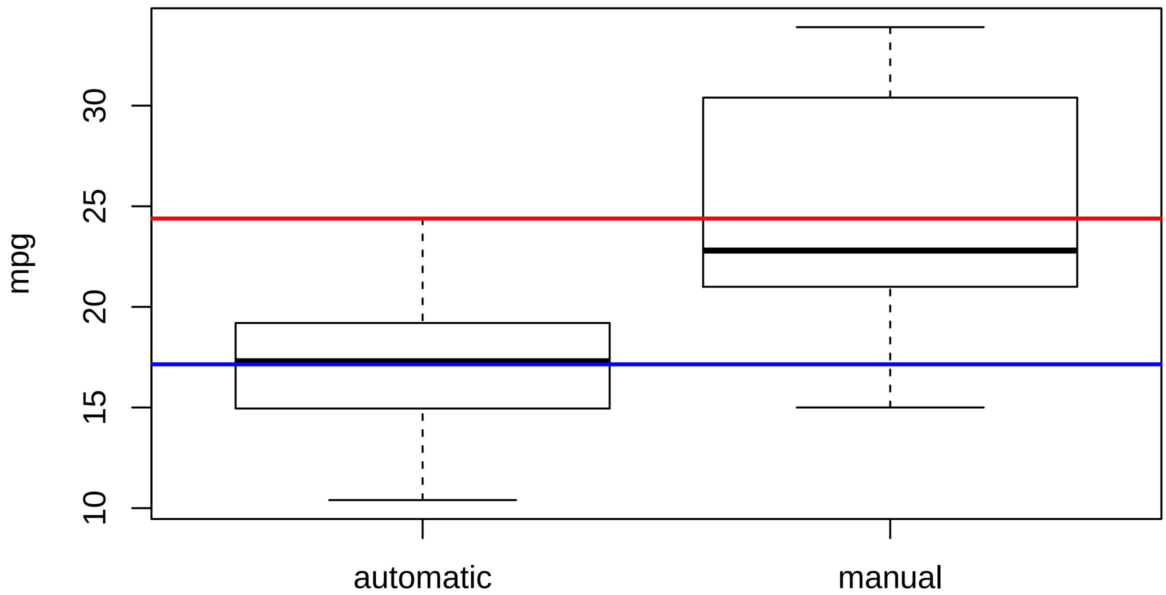
```r
boxplot(mtcars$mpg, mtcars$mpg[mtcars$am == 0], ylab = "mpg", names = c("overall",
    "automobile"))
abline(h = mean(mtcars$mpg), lwd = 2, col = "red")
abline(h = mean(mtcars$mpg[mtcars$am == 0]), lwd = 2, col = "blue")
```




```r
mpg.mu = mean(mtcars$mpg)
mpg_am = mtcars$mpg[mtcars$am == 0]
t.test(mpg_am, mu = mpg.mu)
```

```
##
##  One Sample t-test
##
## data:  mpg_am
## t = -3.3462, df = 18, p-value = 0.003595
## alternative hypothesis: true mean is not equal to 20.09062
## 95 percent confidence interval:
##  15.29946 18.99528
## sample estimates:
## mean of x
##  17.14737
```

```r
boxplot(mtcars$mpg ~ mtcars$am, ylab = "mpg", names = c("automatic", "manual"))
abline(h = mean(mtcars$mpg[mtcars$am == 0]), lwd = 2, col = "blue")
abline(h = mean(mtcars$mpg[mtcars$am == 1]), lwd = 2, col = "red")
```




```
t.test(mtcars$mpg ~ mtcars$am)
```

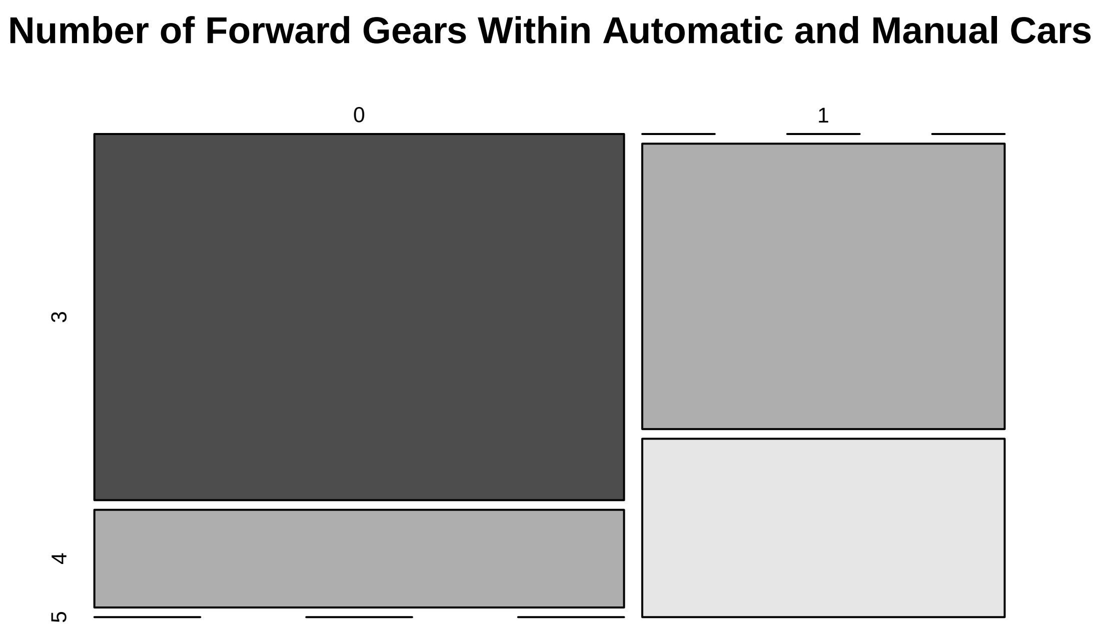
```
##
##  Welch Two Sample t-test
##
## data:  mtcars$mpg by mtcars$am
## t = -3.7671, df = 18.332, p-value = 0.001374
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -11.280194  -3.209684
## sample estimates:
## mean in group 0 mean in group 1
##        17.14737        24.39231
```

### 14.2 Pearson's Chi-squared Test

```
ftable = table(mtcars$am, mtcars$gear)
ftable
```

```
##
##      3  4  5
##   0 15  4  0
##   1  0  8  5
```

```
mosaicplot(ftable, main = "Number of Forward Gears Within Automatic and Manual Cars",
    color = TRUE)
```




```
chisq.test(ftable)
```

```
##
##  Pearson's Chi-squared test
##
## data:  ftable
## X-squared = 20.945, df = 2, p-value = 2.831e-05
```

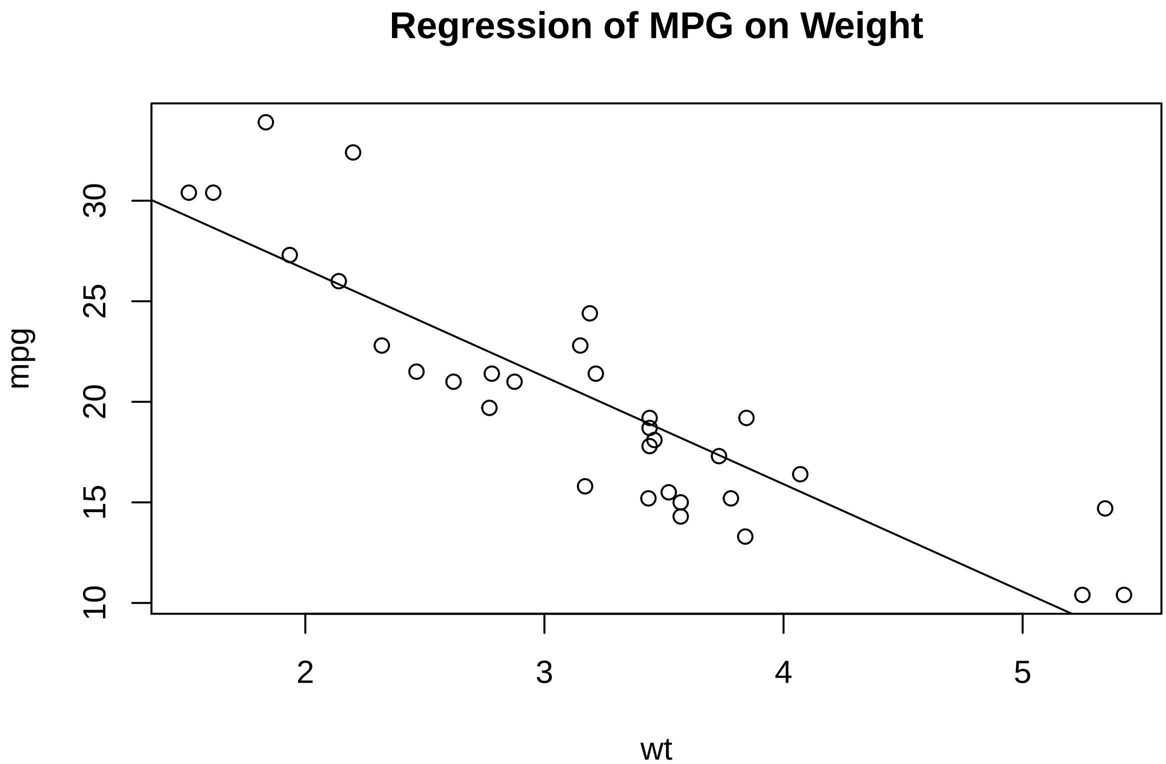
# 15 Creating a Graph

In R, graphs are typically created interactively.

```
# Creating a Graph
attach(mtcars)
```

```
## The following object is masked from package:ggplot2:
##
##     mpg
```

```
plot(wt, mpg)
abline(lm(mpg ~ wt))
title("Regression of MPG on Weight")
```




The plot( ) function opens a graph window and plots weight vs. miles per gallon. The next line of code adds a regression line to this graph. The final line adds a title.

Saving Graphs You can save the graph in a variety of formats from the menu File -> Save As.

You can also save the graph via code using one of the following functions.

Function Output to pdf("mygraph.pdf") pdf file win.metafile("mygraph.wmf") windows metafile png("mygraph.png") png file jpeg("mygraph.jpg") jpeg file bmp("mygraph.bmp") bmp file postscript("mygraph.ps") postscript file See input/output for details.

Viewing Several Graphs Creating a new graph by issuing a high level plotting command (plot, hist, boxplot, etc.) will typically overwrite a previous graph. To avoid this, open a new graph window before creating a new graph. To open a new graph window use one of the functions below.

Function Platform windows() Windows X11() Unix quartz() Mac You can have multiple graph windows open at one time. See help(dev.cur) for more details.

Alternatively, after opening the first graph window, choose History -> Recording from the graph window menu. Then you can use Previous and Next to step through the graphs you have created.

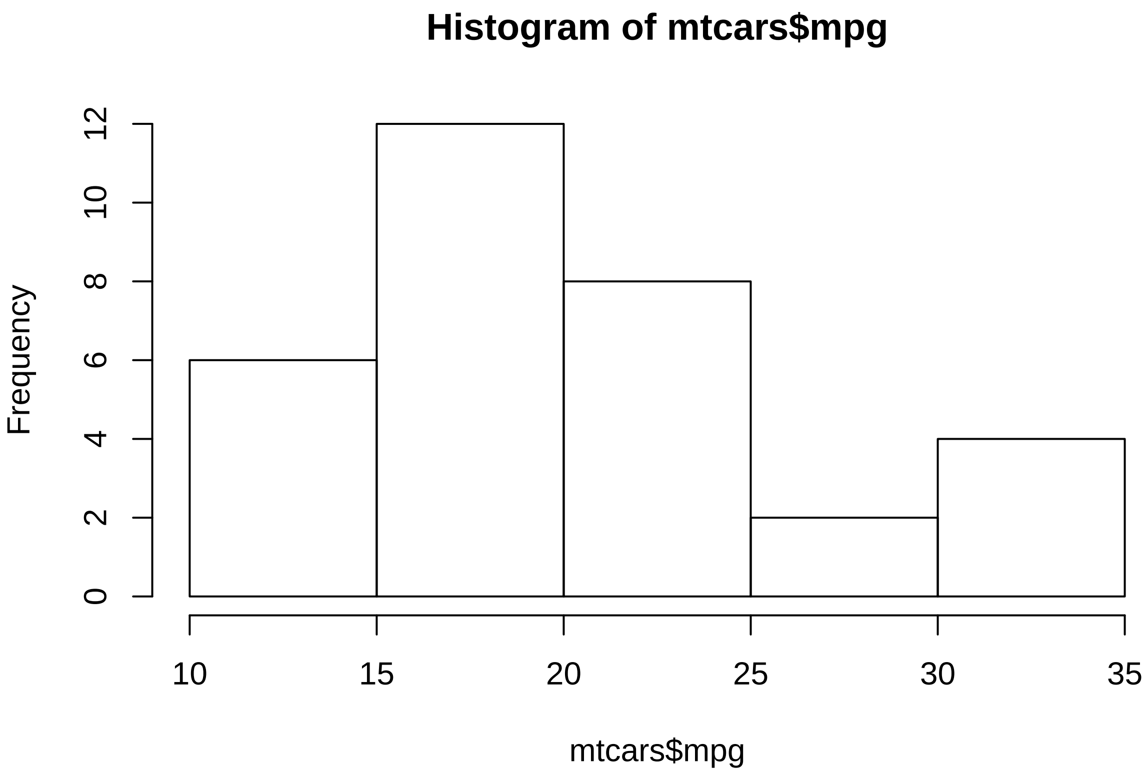
Graphical Parameters You can specify fonts, colors, line styles, axes, reference lines, etc. by specifying graphical parameters. This allows a wide degree of customization. Graphical parameters, are covered in the Advanced Graphs section. The Advanced Graphs section also includes a more detailed coverage of axis and text customization.

# 16 Histograms and Density Plots

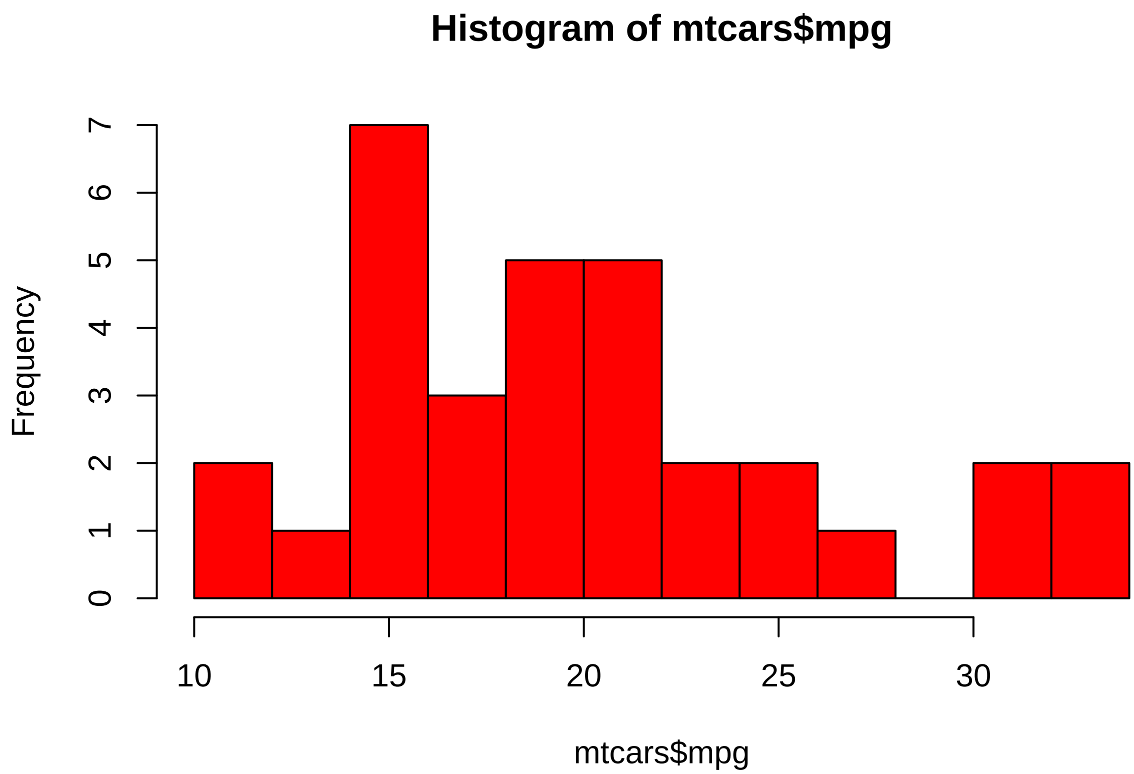
## 16.1 Histograms

You can create histograms with the function hist(x) where x is a numeric vector of values to be plotted. The option freq=FALSE plots probability densities instead of frequencies. The option breaks= controls the number of bins.

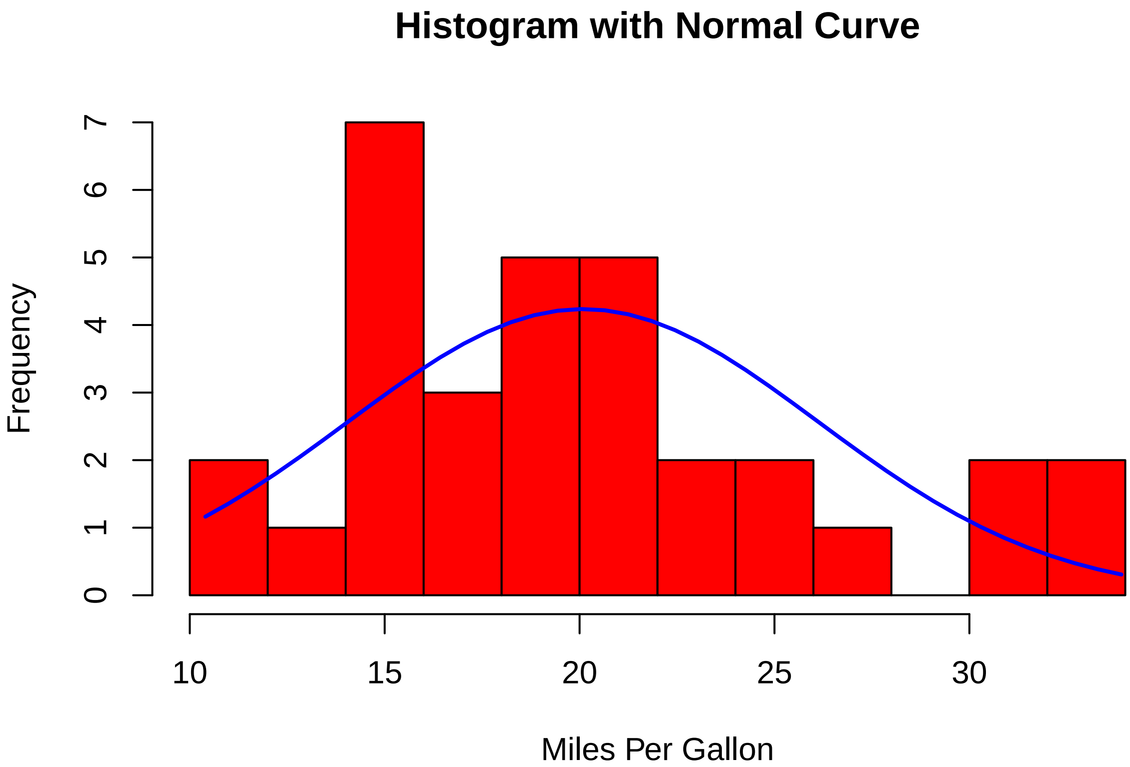
```
# Simple Histogram
hist(mtcars$mpg)
```




```
# Colored Histogram with Different Number of Bins
hist(mtcars$mpg, breaks = 12, col = "red")
```




```r
# Add a Normal Curve (Thanks to Peter Dalgaard)
x <- mtcars$mpg
h <- hist(x, breaks = 10, col = "red", xlab = "Miles Per Gallon", main = "Histogram with Normal Curve")
xfit <- seq(min(x), max(x), length = 40)
yfit <- dnorm(xfit, mean = mean(x), sd = sd(x))
yfit <- yfit * diff(h$mids[1:2]) * length(x)
lines(xfit, yfit, col = "blue", lwd = 2)
```



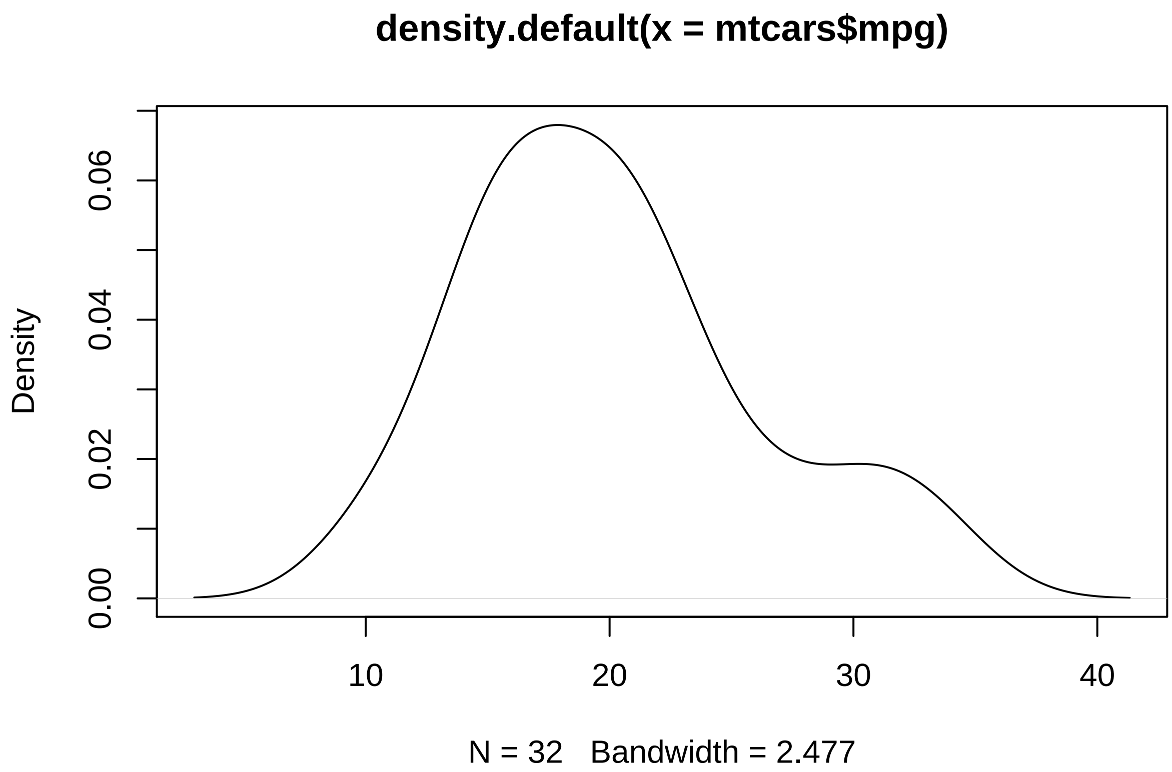

Histograms can be a poor method for determining the shape of a distribution because it is so strongly affected by the number of bins used.

To practice making a density plot with the hist() function, try this exercise.

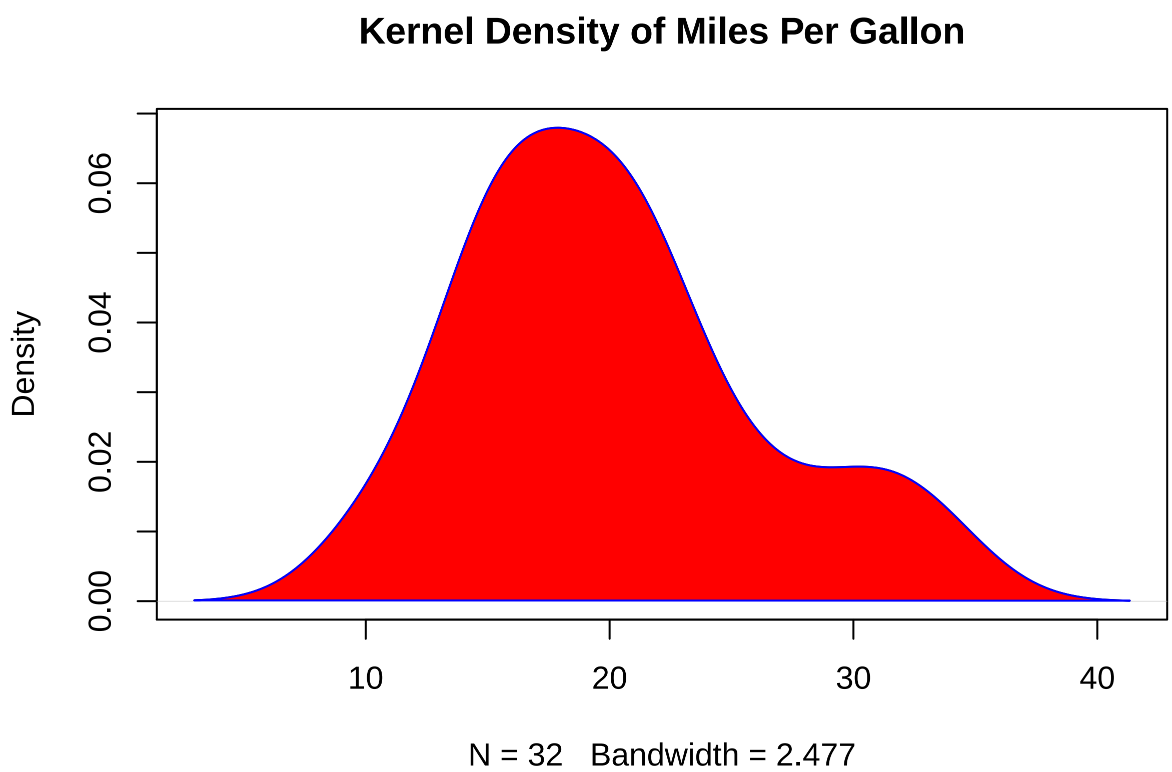
## 16.2 Kernel Density Plots

Kernal density plots are usually a much more effective way to view the distribution of a variable. Create the plot using plot(density(x)) where x is a numeric vector.

```
# Kernel Density Plot
d <- density(mtcars$mpg)  # returns the density data
plot(d)  # plots the results
```




```r
# Filled Density Plot
d <- density(mtcars$mpg)
plot(d, main = "Kernel Density of Miles Per Gallon")
polygon(d, col = "red", border = "blue")
```

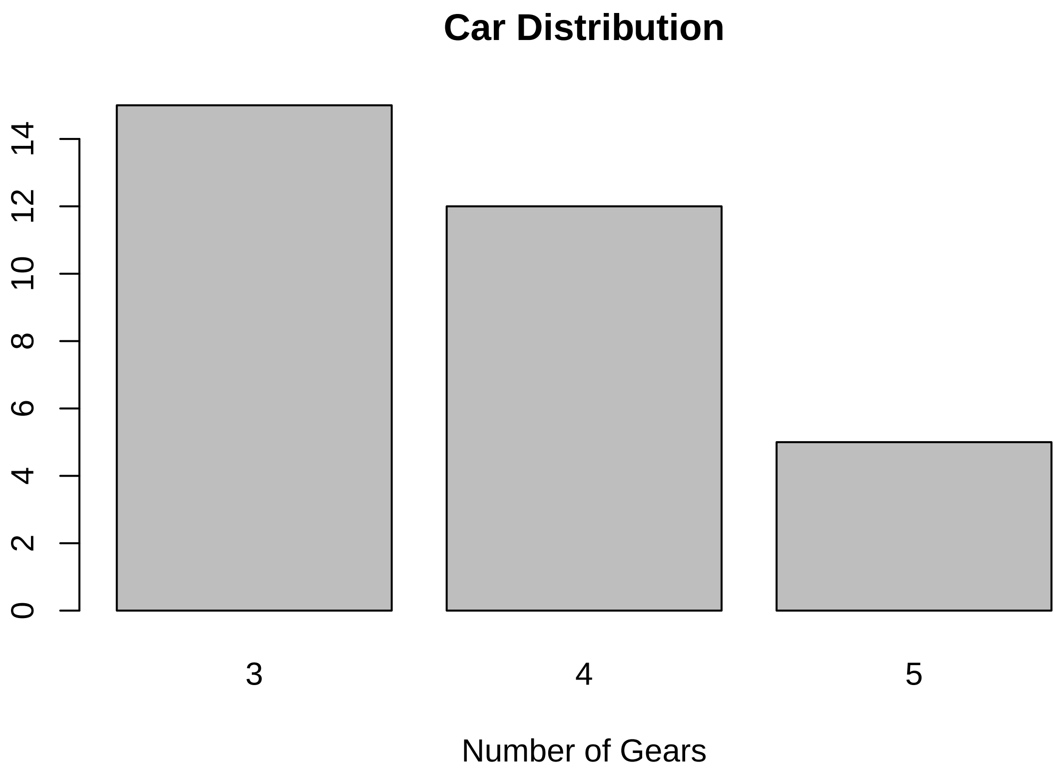


# 17 Bar Plots

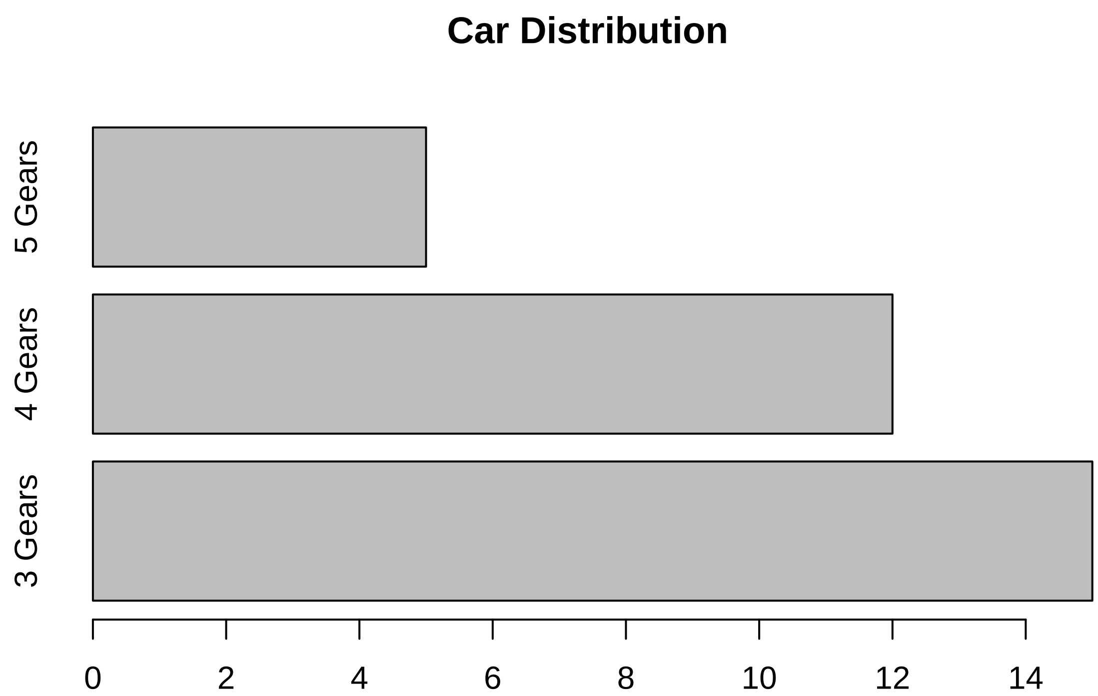
Create barplots with the barplot(height) function, where height is a vector or matrix. If height is a vector, the values determine the heights of the bars in the plot. If height is a matrix and the option beside=FALSE then each bar of the plot corresponds to a column of height, with the values in the column giving the heights of stacked "sub-bars". If height is a matrix and beside=TRUE, then the values in each column are juxtaposed rather than stacked. Include option names.arg=(character vector) to label the bars. The option horiz=TRUE to createa a horizontal barplot.

## 17.1 Simple Bar Plot

```
# Simple Bar Plot
counts <- table(mtcars$gear)
barplot(counts, main = "Car Distribution", xlab = "Number of Gears")
```



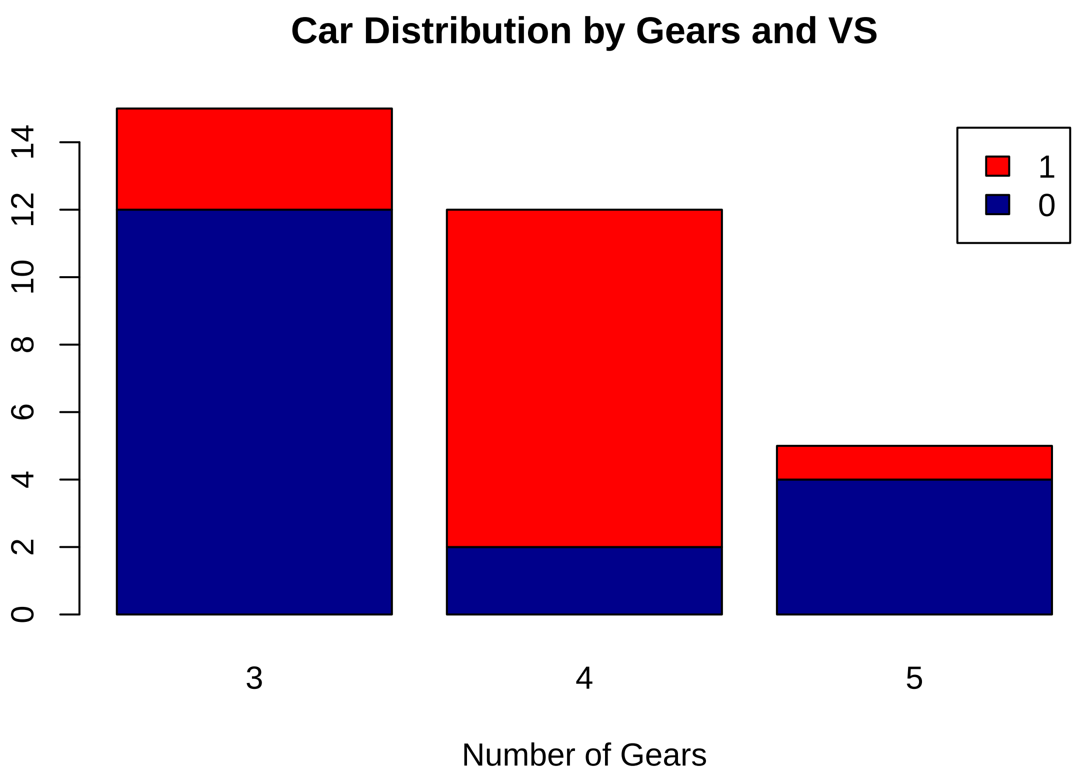

```
# Simple Horizontal Bar Plot with Added Labels
counts <- table(mtcars$gear)
barplot(counts, main = "Car Distribution", horiz = TRUE, names.arg = c("3 Gears",
    "4 Gears", "5 Gears"))
```




### 17.2 Stacked Bar Plot

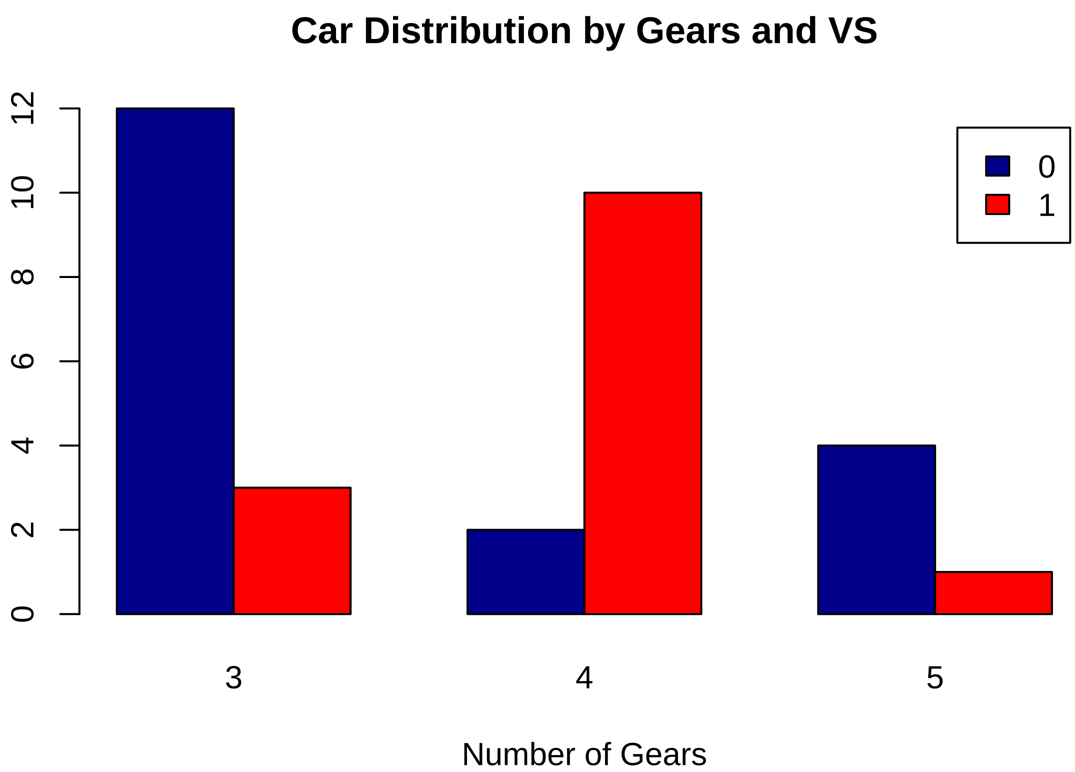
# 18 Stacked Bar Plot with Colors and Legend

```
counts <- table(mtcars$vs, mtcars$gear)
barplot(counts, main = "Car Distribution by Gears and VS", xlab = "Number of Gears",
    col = c("darkblue", "red"), legend = rownames(counts))
```




## 18.1 Grouped Bar Plot

```
# Grouped Bar Plot
counts <- table(mtcars$vs, mtcars$gear)
barplot(counts, main = "Car Distribution by Gears and VS", xlab = "Number of Gears",
    col = c("darkblue", "red"), legend = rownames(counts), beside = TRUE)
```



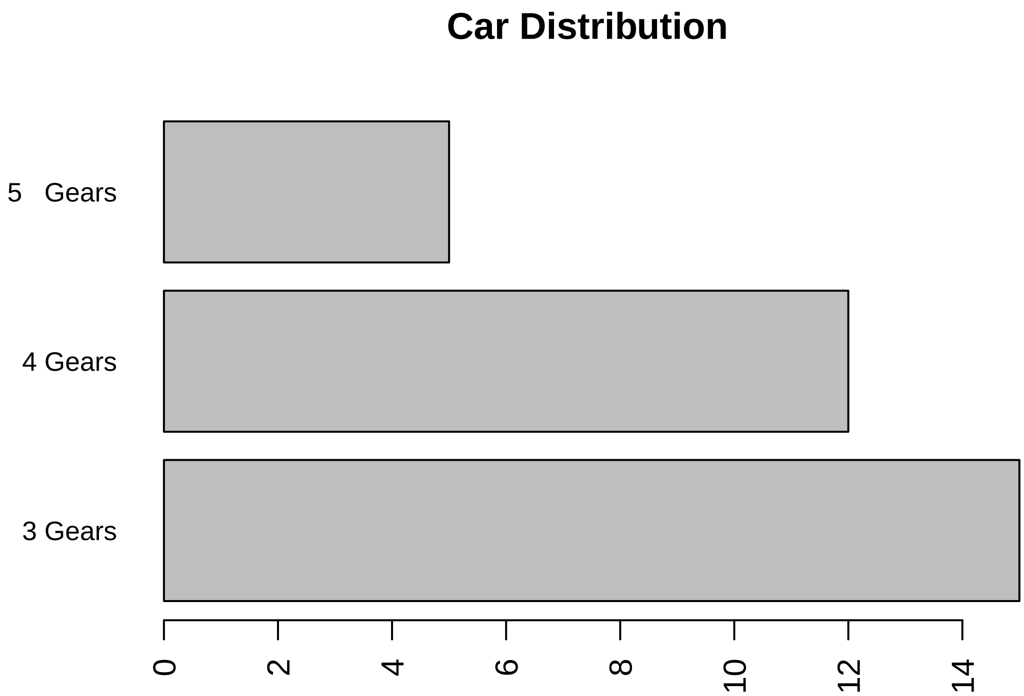

## 18.2 Notes

Bar plots need not be based on counts or frequencies. You can create bar plots that represent means, medians, standard deviations, etc. Use the aggregate( ) function and pass the results to the barplot( ) function.

By default, the categorical axis line is suppressed. Include the option axis.lty=1 to draw it.

With many bars, bar labels may start to overlap. You can decrease the font size using the cex.names = option. Values smaller than one will shrink the size of the label. Additionally, you can use graphical parameters such as the following to help text spacing:

```
# Fitting Labels
par(las = 2)  # make label text perpendicular to axis
par(mar = c(5, 8, 4, 2))  # increase y-axis margin.

counts <- table(mtcars$gear)
barplot(counts, main = "Car Distribution", horiz = TRUE, names.arg = c("3 Gears",
    "4 Gears", "5   Gears"), cex.names = 0.8)
```

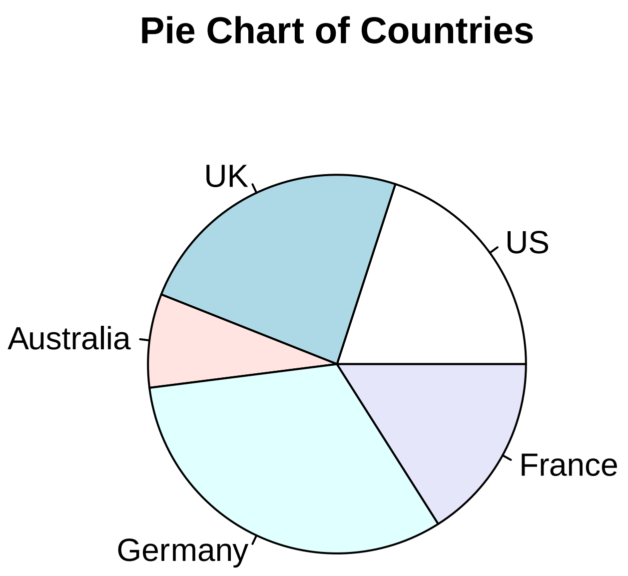



## 19 Pie Charts

Pie charts are not recommended in the R documentation, and their features are somewhat limited. The authors recommend bar or dot plots over pie charts because people are able to judge length more accurately than volume. Pie charts are created with the function pie(x, labels=) where x is a non-negative numeric vector indicating the area of each slice and labels= notes a character vector of names for the slices.

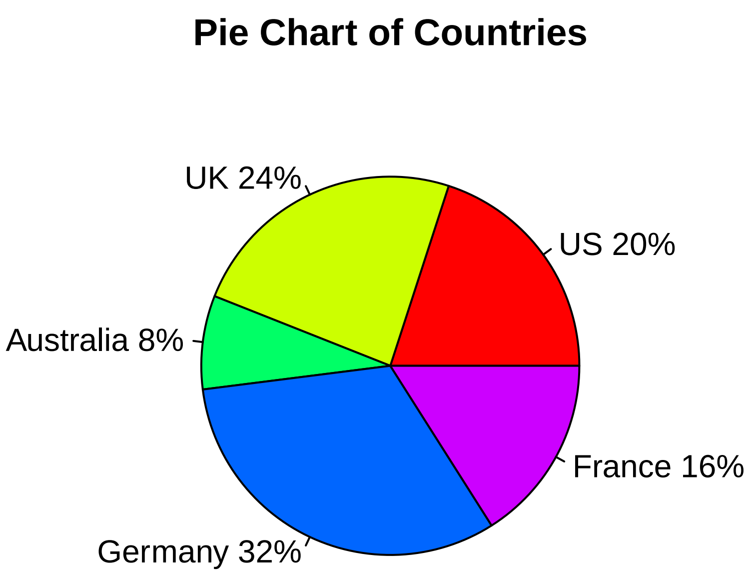
### 19.1 Simple Pie Chart

```
# Simple Pie Chart
slices <- c(10, 12, 4, 16, 8)
lbls <- c("US", "UK", "Australia", "Germany", "France")
pie(slices, labels = lbls, main = "Pie Chart of Countries")
```




## 19.2 Pie Chart with Annotated Percentages

```
# Pie Chart with Percentages
slices <- c(10, 12, 4, 16, 8)
lbls <- c("US", "UK", "Australia", "Germany", "France")
pct <- round(slices/sum(slices) * 100)
lbls <- paste(lbls, pct)  # add percents to labels
lbls <- paste(lbls, "%", sep = "")  # ad % to labels
pie(slices, labels = lbls, col = rainbow(length(lbls)), main = "Pie Chart of Countries")
```



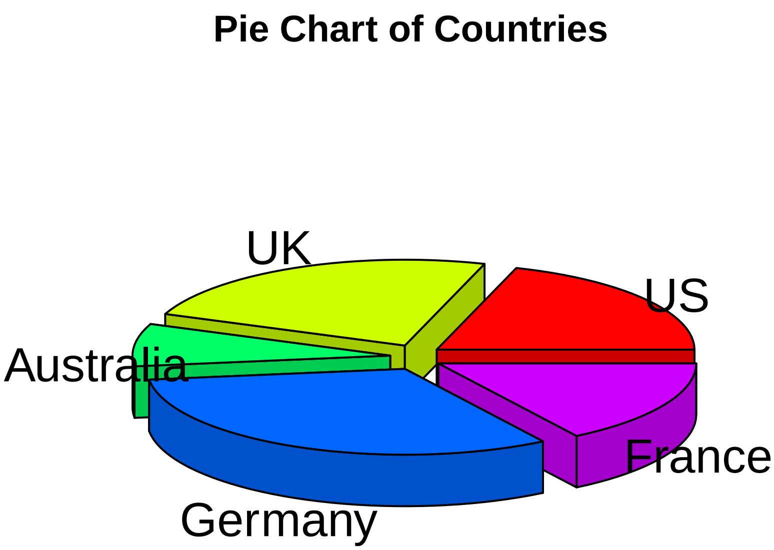

## 19.3 3D Pie Chart

The pie3D( ) function in the plotrix package provides 3D exploded pie charts.

```r
# 3D Exploded Pie Chart
library(plotrix)
```

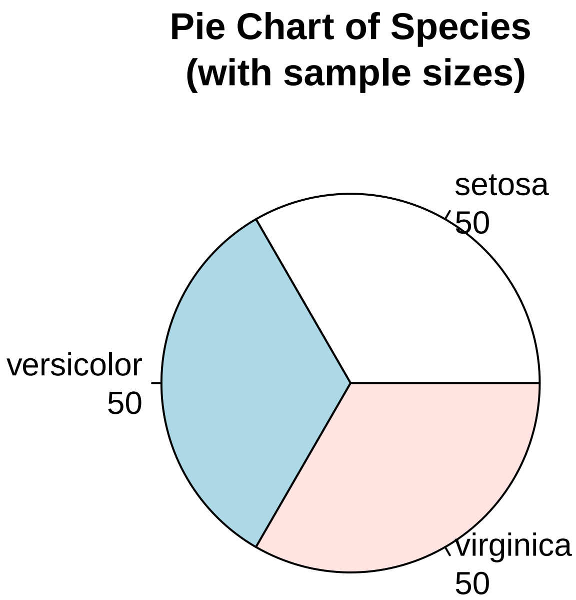
```
##
## Attaching package: 'plotrix'

## The following object is masked from 'package:NMF':
##
##     dispersion
```

```r
slices <- c(10, 12, 4, 16, 8)
lbls <- c("US", "UK", "Australia", "Germany", "France")
pie3D(slices, labels = lbls, explode = 0.1, main = "Pie Chart of Countries ")
```




### 19.4 Creating Annotated Pies from a data frame

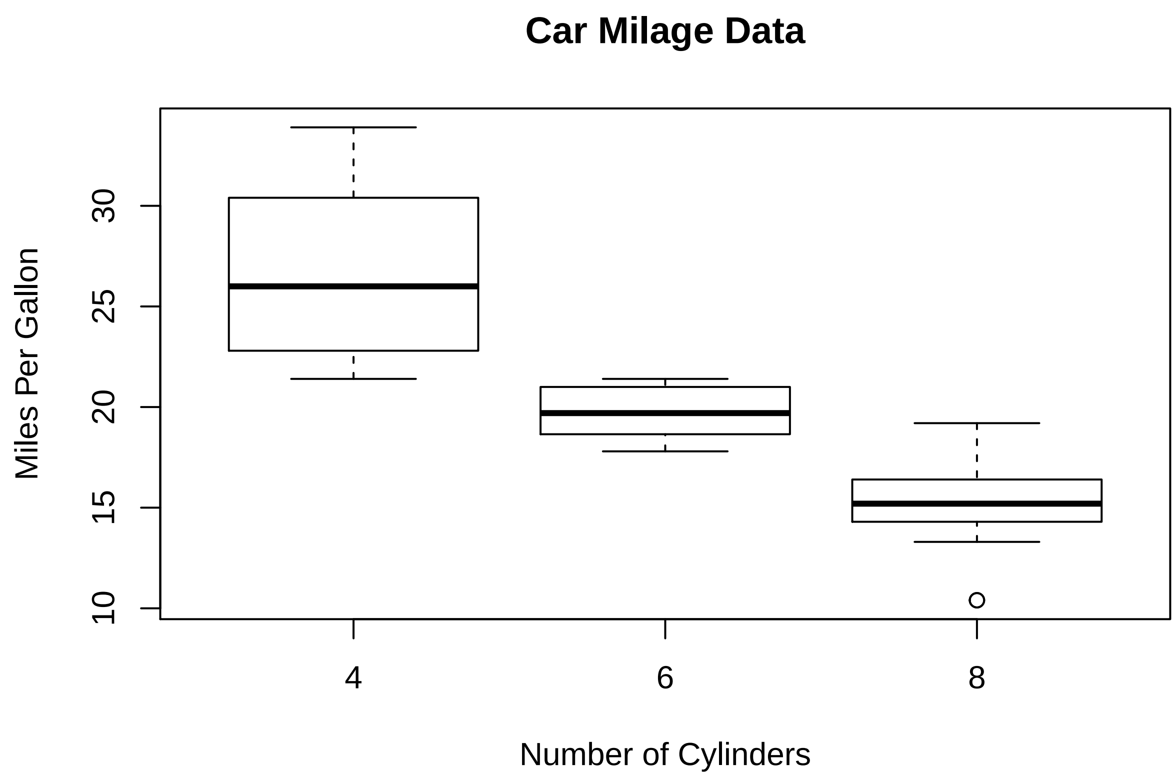
```r
# Pie Chart from data frame with Appended Sample Sizes
mytable <- table(iris$Species)
lbls <- paste(names(mytable), "\n", mytable, sep = "")
pie(mytable, labels = lbls, main = "Pie Chart of Species\n (with sample sizes)")
```



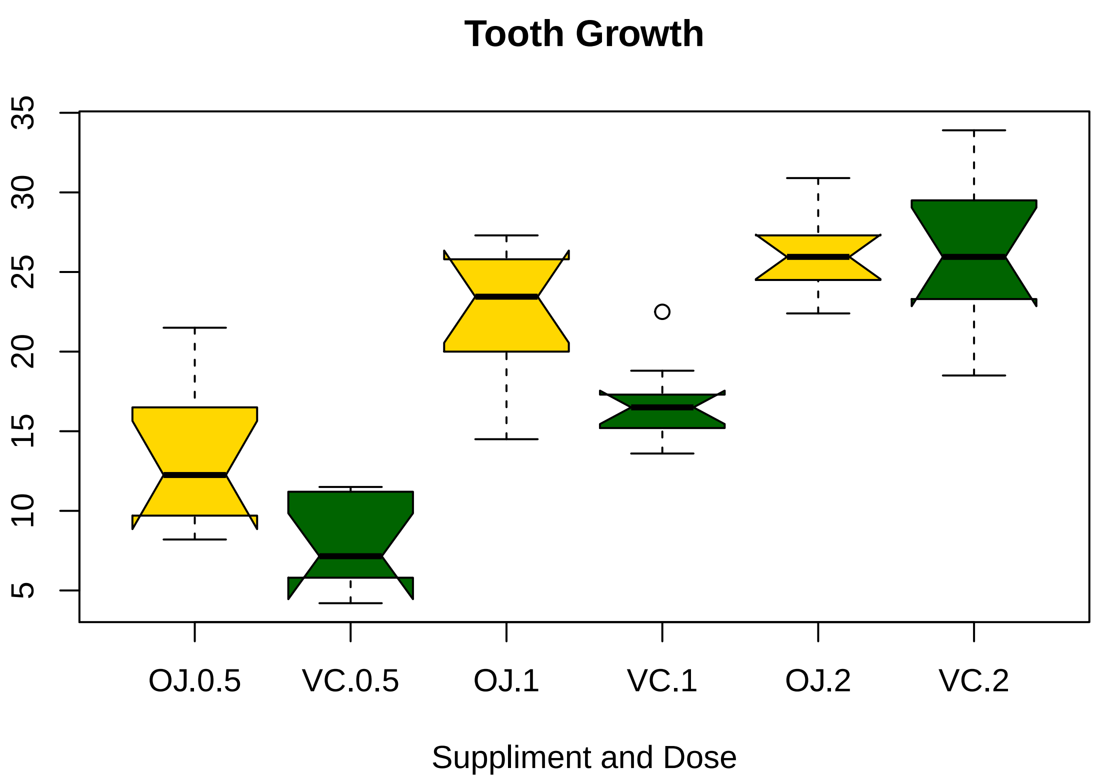

## 20 Boxplots

Boxplots can be created for individual variables or for variables by group. The format is boxplot(x, data=), where x is a formula and data= denotes the data frame providing the data. An example of a formula is y~group where a separate boxplot for numeric variable y is generated for each value of group. Add varwidth=TRUE to make boxplot widths proportional to the square root of the samples sizes. Add horizontal=TRUE to reverse the axis orientation.

```r
# Boxplot of MPG by Car Cylinders
boxplot(mpg ~ cyl, data = mtcars, main = "Car Milage Data", xlab = "Number of Cylinders",
    ylab = "Miles Per Gallon")
```




```r
# Notched Boxplot of Tooth Growth Against 2 Crossed Factors boxes colored
# for ease of interpretation
boxplot(len ~ supp * dose, data = ToothGrowth, notch = TRUE, col = (c("gold",
    "darkgreen")), main = "Tooth Growth", xlab = "Suppliment and Dose")
```



In the notched boxplot, if two boxes' notches do not overlap this is 'strong evidence' their medians differ (Chambers et al., 1983, p. 62).

Colors recycle. In the example above, if I had listed 6 colors, each box would have its own color. Earl F. Glynn has created an easy to use list of colors is PDF format.

## 20.1 Other Options

The boxplot.matrix( ) function in the sfsmisc package draws a boxplot for each column (row) in a matrix. The boxplot.n( ) function in the gplots package annotates each boxplot with its sample size. The bplot( ) function in the Rlab package offers many more options controlling the positioning and labeling of boxes in the output.

## 20.2 Violin Plots

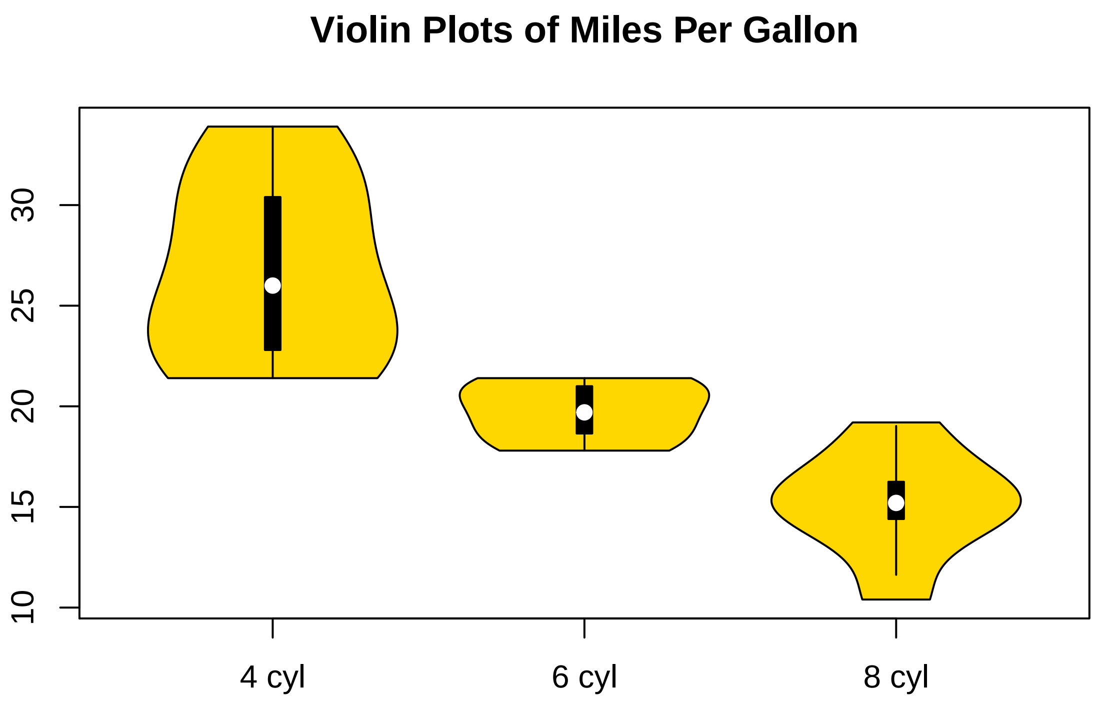
A violin plot is a combination of a boxplot and a kernel density plot. They can be created using the vioplot( ) function from vioplot package.

```
# Violin Plots
library(vioplot)
```

```
## Loading required package: sm
```

```
## Package 'sm', version 2.2-5.4: type help(sm) for summary information
```

```
x1 <- mtcars$mpg[mtcars$cyl == 4]
x2 <- mtcars$mpg[mtcars$cyl == 6]
x3 <- mtcars$mpg[mtcars$cyl == 8]
vioplot(x1, x2, x3, names = c("4 cyl", "6 cyl", "8 cyl"), col = "gold")
title("Violin Plots of Miles Per Gallon")
```



# 21 Scatterplots

## 21.1 Simple Scatterplot

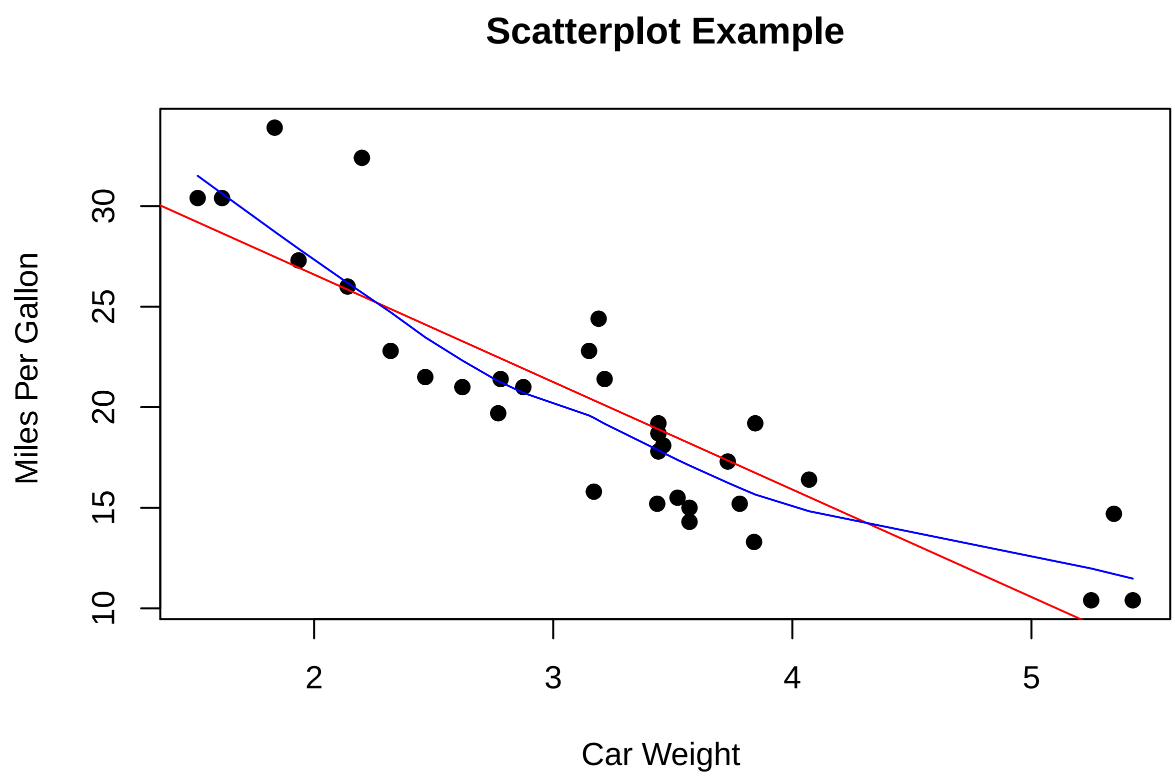
There are many ways to create a scatterplot in R. The basic function is plot(x, y), where x and y are numeric vectors denoting the (x,y) points to plot.

```
# Simple Scatterplot
attach(mtcars)
```

```
## The following objects are masked from mtcars (pos = 6):
##
##     am, carb, cyl, disp, drat, gear, hp, mpg, qsec, vs, wt
```

```
## The following object is masked from package:ggplot2:
##
##     mpg
```

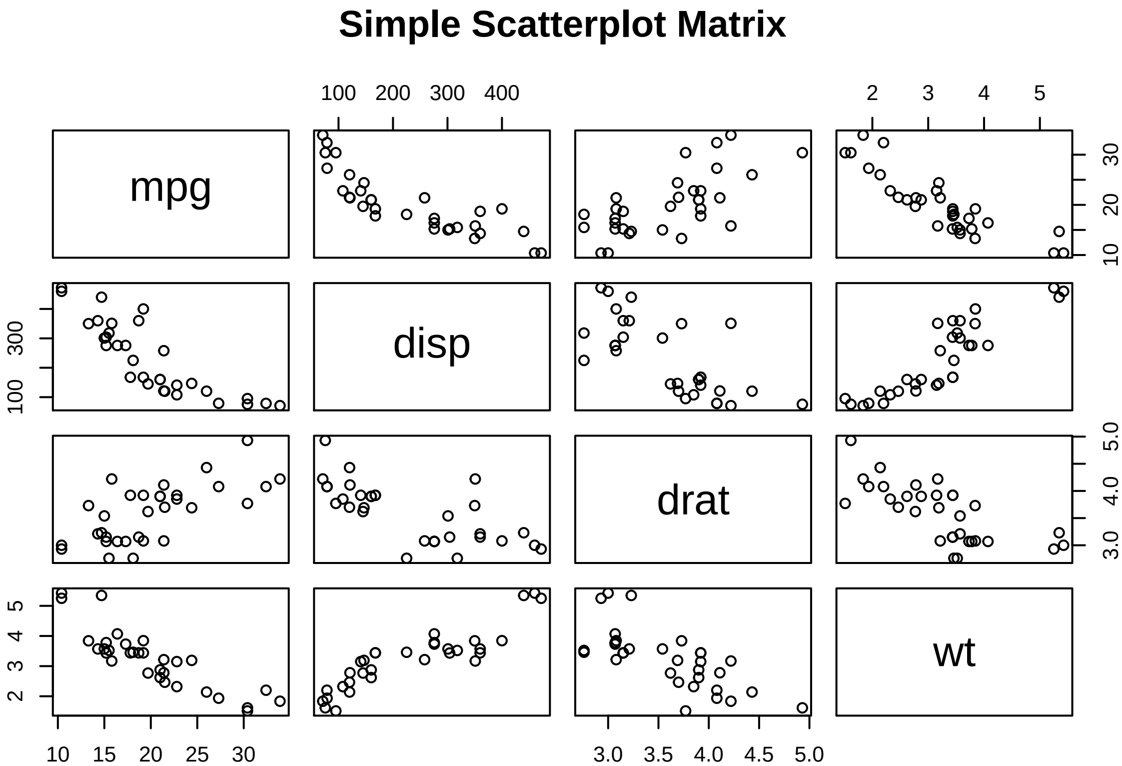
```
plot(wt, mpg, main = "Scatterplot Example", xlab = "Car Weight ", ylab = "Miles Per Gallon ",
    pch = 19)


# Add fit lines
abline(lm(mpg ~ wt), col = "red")  # regression line (y~x)
lines(lowess(wt, mpg), col = "blue")  # lowess line (x,y)
```




## 21.2 Scatterplot Matrices

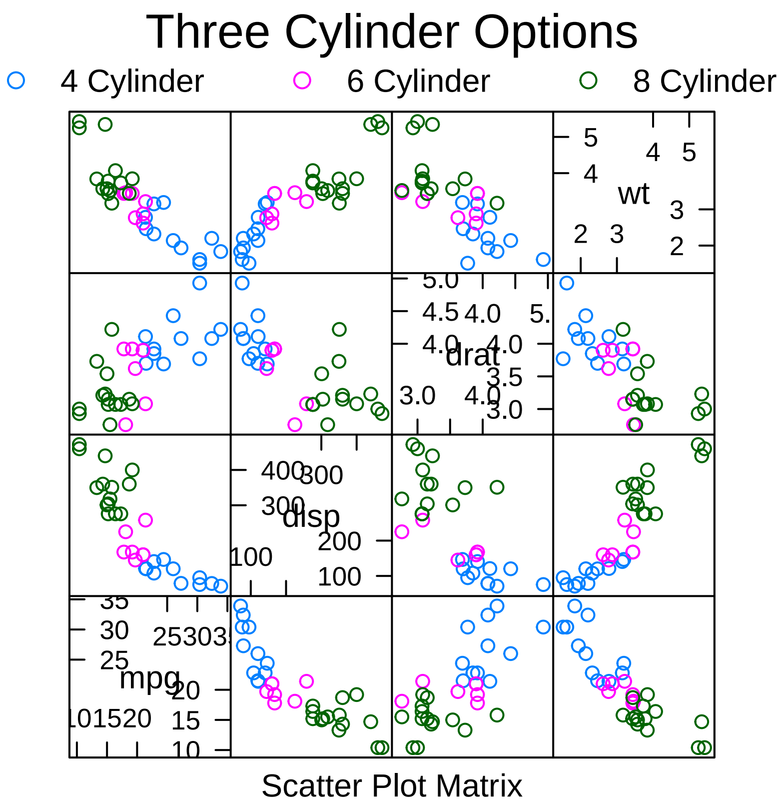
There are at least 4 useful functions for creating scatterplot matrices. Analysts must love scatterplot matrices!

```r
# Basic Scatterplot Matrix
pairs(~mpg + disp + drat + wt, data = mtcars, main = "Simple Scatterplot Matrix")
```




The lattice package provides options to condition the scatterplot matrix on a factor.

```r
# Scatterplot Matrices from the lattice Package
library(lattice)
super.sym <- trellis.par.get("superpose.symbol")
splom(mtcars[c(1, 3, 5, 6)], groups = cyl, data = mtcars, panel = panel.superpose,
    key = list(title = "Three Cylinder Options", columns = 3, points = list(pch = super.sym$pch[1:3],
        col = super.sym$col[1:3]), text = list(c("4 Cylinder", "6 Cylinder",
        "8 Cylinder"))))
```

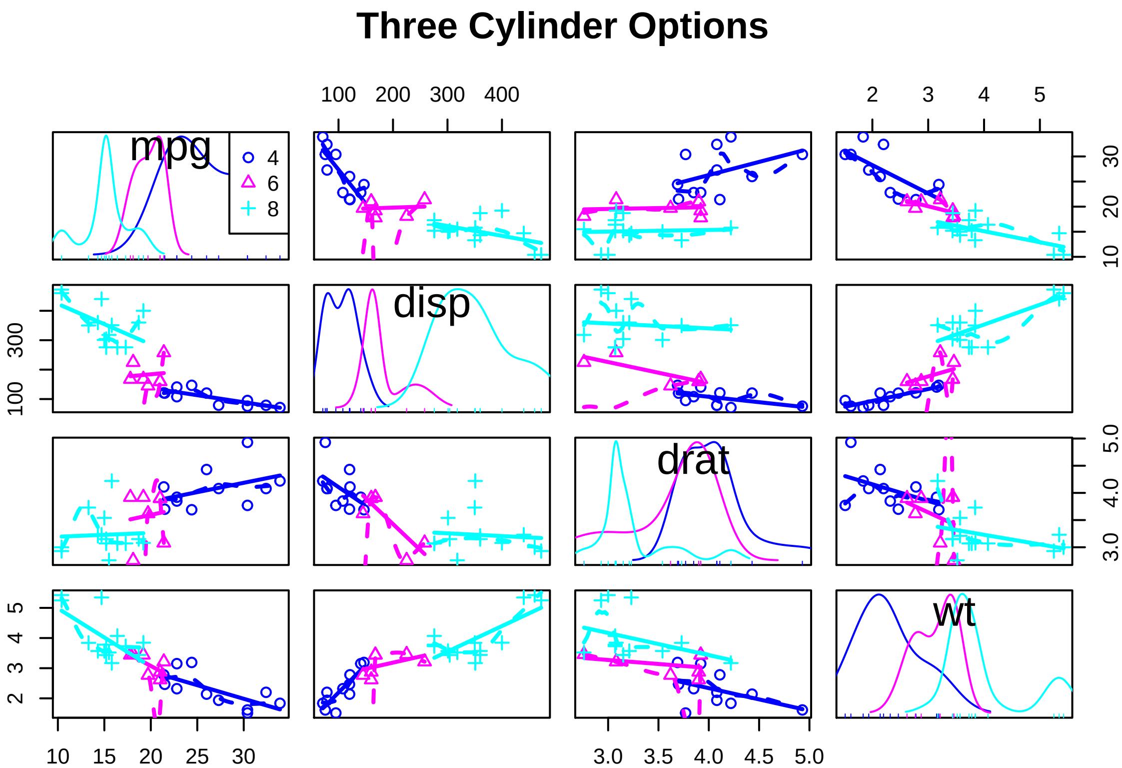



The car package can condition the scatterplot matrix on a factor, and optionally include lowess and linear best fit lines, and boxplot, densities, or histograms in the principal diagonal, as well as rug plots in the margins of the cells.

```
# Scatterplot Matrices from the car Package
library(car)
```

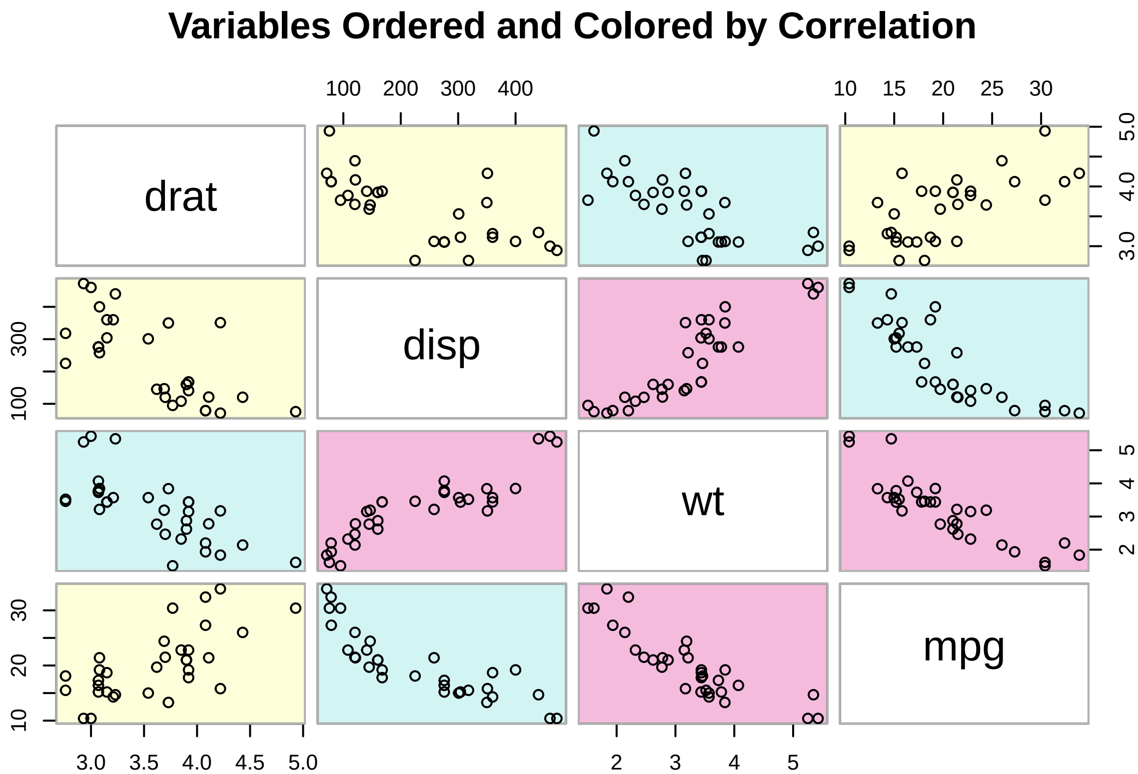
```
## Loading required package: carData
```

```
scatterplotMatrix(~mpg + disp + drat + wt | cyl, data = mtcars, main = "Three Cylinder Options")
```




The gclus package provides options to rearrange the variables so that those with higher correlations are closer to the principal diagonal. It can also color code the cells to reflect the size of the correlations.

```
# Scatterplot Matrices from the glus Package
library(gclus)
dta <- mtcars[c(1, 3, 5, 6)]  # get data
dta.r <- abs(cor(dta))  # get correlations
dta.col <- dmat.color(dta.r)  # get colors
# reorder variables so those with highest correlation are closest to the
# diagonal
dta.o <- order.single(dta.r)
cpairs(dta, dta.o, panel.colors = dta.col, gap = 0.5, main = "Variables Ordered and Colored by Correlati
```



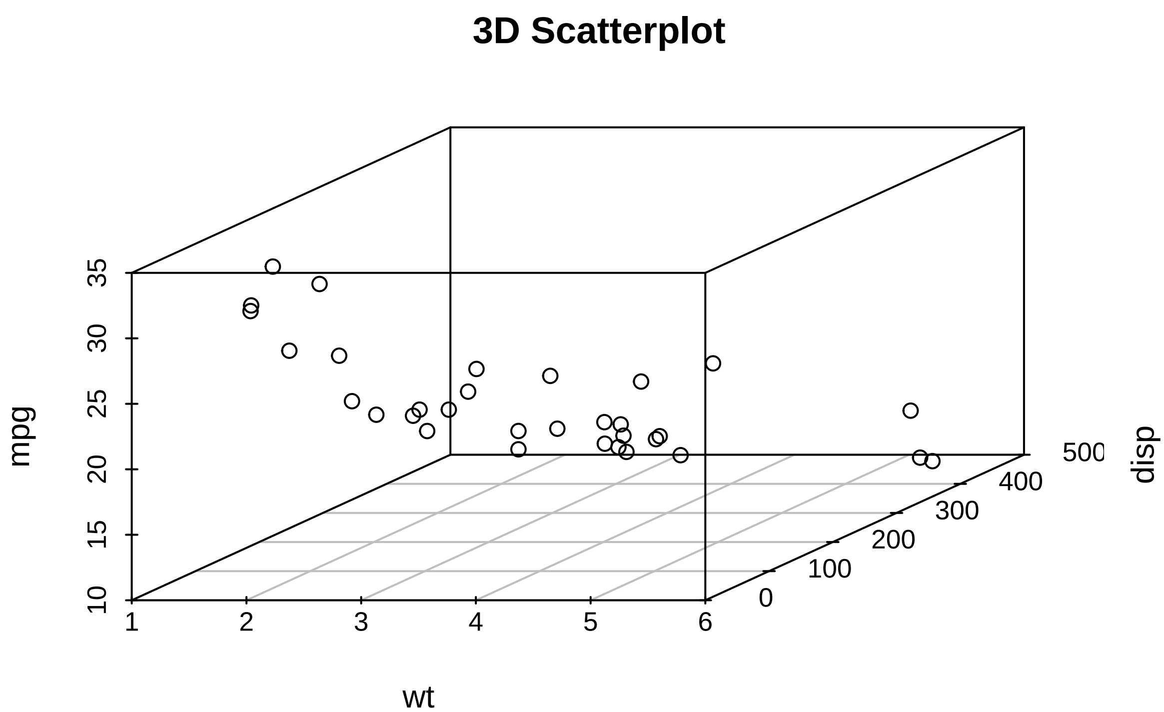

## 21.3 3D Scatterplots

You can create a 3D scatterplot with the scatterplot3d package. Use the function scatterplot3d(x, y, z).

```
# 3D Scatterplot
library(scatterplot3d)
attach(mtcars)
```

```
## The following objects are masked from mtcars (pos = 8):
##
##     am, carb, cyl, disp, drat, gear, hp, mpg, qsec, vs, wt

## The following objects are masked from mtcars (pos = 12):
##
##     am, carb, cyl, disp, drat, gear, hp, mpg, qsec, vs, wt

## The following object is masked from package:ggplot2:
##
##     mpg
```

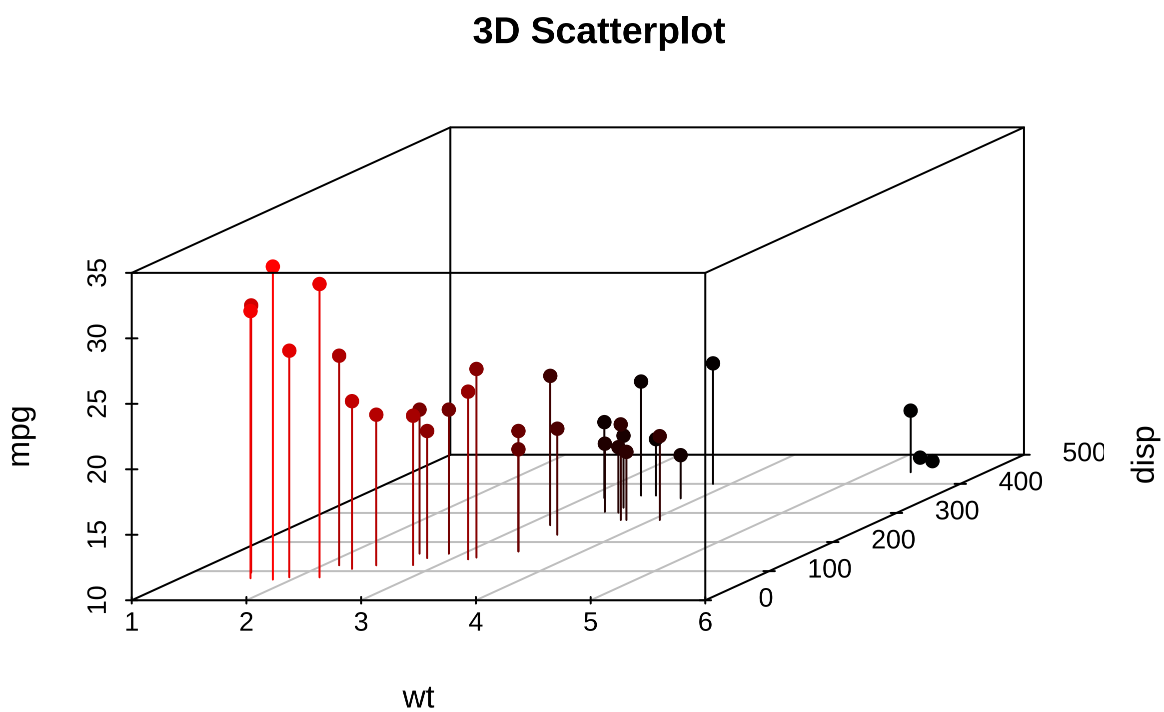
```
scatterplot3d(wt, disp, mpg, main = "3D Scatterplot")
```




```
# 3D Scatterplot with Coloring and Vertical Drop Lines
library(scatterplot3d)
attach(mtcars)
```

```
## The following objects are masked from mtcars (pos = 3):
##
##     am, carb, cyl, disp, drat, gear, hp, mpg, qsec, vs, wt

## The following objects are masked from mtcars (pos = 9):
##
##     am, carb, cyl, disp, drat, gear, hp, mpg, qsec, vs, wt

## The following objects are masked from mtcars (pos = 13):
##
##     am, carb, cyl, disp, drat, gear, hp, mpg, qsec, vs, wt

## The following object is masked from package:ggplot2:
##
##     mpg
```

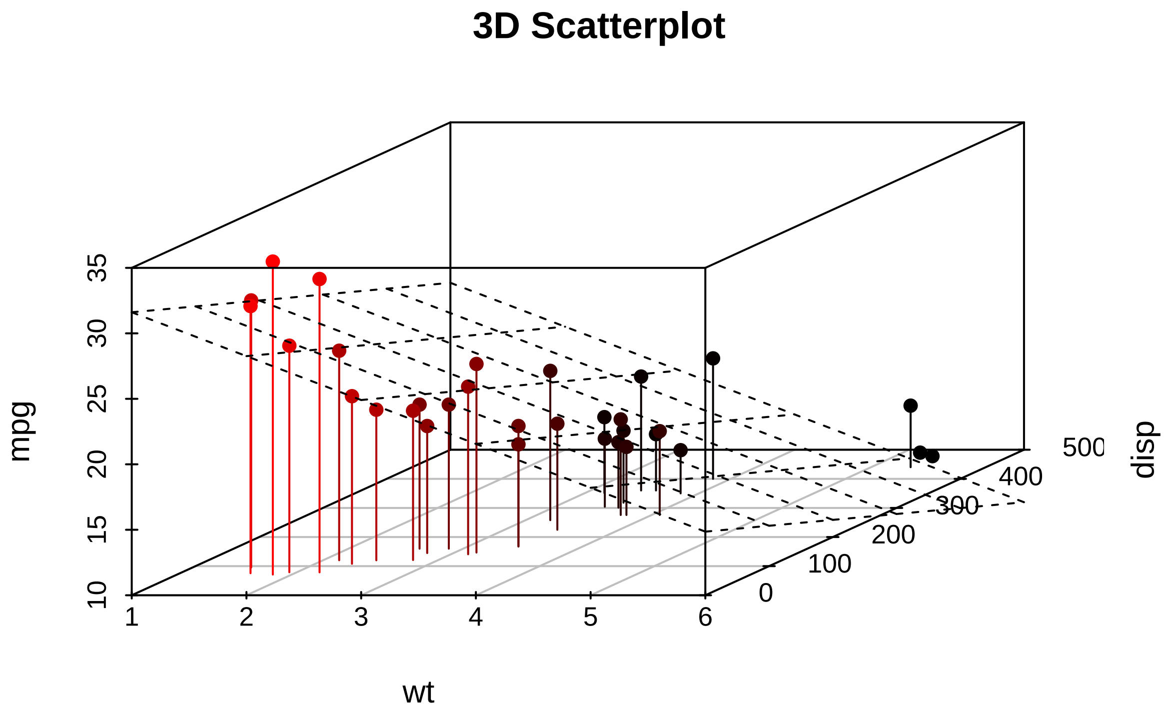
```
scatterplot3d(wt, disp, mpg, pch = 16, highlight.3d = TRUE, type = "h", main = "3D Scatterplot")
```




```
# 3D Scatterplot with Coloring and Vertical Lines and Regression Plane
library(scatterplot3d)
attach(mtcars)
```

```
## The following objects are masked from mtcars (pos = 3):
##
##     am, carb, cyl, disp, drat, gear, hp, mpg, qsec, vs, wt

## The following objects are masked from mtcars (pos = 4):
##
##     am, carb, cyl, disp, drat, gear, hp, mpg, qsec, vs, wt

## The following objects are masked from mtcars (pos = 10):
##
##     am, carb, cyl, disp, drat, gear, hp, mpg, qsec, vs, wt

## The following objects are masked from mtcars (pos = 14):
##
##     am, carb, cyl, disp, drat, gear, hp, mpg, qsec, vs, wt

## The following object is masked from package:ggplot2:
##
##     mpg
```

```
s3d <- scatterplot3d(wt, disp, mpg, pch = 16, highlight.3d = TRUE, type = "h",
    main = "3D Scatterplot")
fit <- lm(mpg ~ wt + disp)
s3d$plane3d(fit)
```




## 21.4 Spinning 3D Scatterplots

You can also create an interactive 3D scatterplot using the plot3D(x, y, z) function in the rgl package. It creates a spinning 3D scatterplot that can be rotated with the mouse. The first three arguments are the x, y, and z numeric vectors representing points. col= and size= control the color and size of the points respectively.

```
# Spinning 3d Scatterplot
library(rgl)

plot3d(wt, disp, mpg, col = "red", size = 3)
```

You can perform a similar function with the scatter3d(x, y, z) in the Rcmdr package.

```
# Another Spinning 3d Scatterplot
library(Rcmdr)
attach(mtcars)
scatter3d(wt, disp, mpg)
```