Data Mining Practice Part.2

1. Data Import

This function reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fileds in the file.

1.1. read.table()

Usage

```
  read.table(file, header = FALSE, sep = "", quote = "\"'",
        dec = ".", row.names, col.names,
        as.is = !stringsAsFactors,
        na.strings = "NA", colClasses = NA, nrows = -1,
        skip = 0, check.names = TRUE, fill = !blank.lines.skip,
        strip.white = FALSE, blank.lines.skip = TRUE,
        comment.char = "#",
        allowEscapes = FALSE, flush = FALSE,
        stringsAsFactors = default.stringsAsFactors(),
        fileEncoding = "", encoding = "unknown", text)
```

```
setwd("C:/Users/dox/Desktop/DM 2013-2/2013-2 데이터마이닝/09.10 실습")

ir.1 <- read.table("C:/Users/dox/Desktop/DM 2013-2/2013-2 데이터마이닝/09.1
0 실습/iris.txt",
    header = T, sep = "\t")
ir.2 <- read.table("C:\\Users\\dox/Desktop\\DM 2013-2\\2013-2 데이터마이닝
\\09.10 실습\\iris.txt",
    header = T, sep = "\t")
ir.3 <- read.table("iris.txt", header = T, sep = "\t")
```

1.2. read.csv()

Usage

```
  read.csv(file, header = TRUE, sep = ",", quote = "\"",
     dec = ".", fill = TRUE, comment.char = "", ...)
```

```
mkt.1 <- read.csv("C:/Users/dox/Desktop/DM 2013-2/2013-2 데이터마이닝/09.10
 실습/Car.csv",
    header = T, sep = ",")
mkt.2 <- read.csv("C:\\Users\\dox/Desktop\\DM 2013-2\\2013-2 데이터마이닝
\\09.10 실습\\Car.csv",
    header = T, sep = ",")
mkt.3 <- read.csv("Car.csv", header = T, sep = ",")
```

1.3. Download Data File from the Web: download.file()

Usage

```
  download.file(url, destfile, method, quiet = FALSE, mode = "w",
          cacheOK = TRUE,
          extra = getOption("download.file.extra"))
```

This function can be used to download a file from Web.

```
# URL
url <- "http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.data"
# Data 가 있는 UCI URL 주소

download.file(url, destfile = "german.data")
# 현재 working directory 로 데이터 download!

german <- read.table("german.data")
# download 받은 데이터 불러오기

head(german, 10)
```

```
##     V1 V2  V3  V4   V5  V6  V7 V8  V9  V10 V11  V12 V13  V14  V15 V16
V17
## 1  A11  6 A34 A43 1169 A65 A75  4 A93 A101   4 A121  67 A143 A152   2 A
173
## 2  A12 48 A32 A43 5951 A61 A73  2 A92 A101   2 A121  22 A143 A152   1 A
173
## 3  A14 12 A34 A46 2096 A61 A74  2 A93 A101   3 A121  49 A143 A152   1 A
172
## 4  A11 42 A32 A42 7882 A61 A74  2 A93 A103   4 A122  45 A143 A153   1 A
173
## 5  A11 24 A33 A40 4870 A61 A73  3 A93 A101   4 A124  53 A143 A153   2 A
173
## 6  A14 36 A32 A46 9055 A65 A73  2 A93 A101   4 A124  35 A143 A153   1 A
172
## 7  A14 24 A32 A42 2835 A63 A75  3 A93 A101   4 A122  53 A143 A152   1 A
173
## 8  A12 36 A32 A41 6948 A61 A73  2 A93 A101   2 A123  35 A143 A151   1 A
174
## 9  A14 12 A32 A43 3059 A64 A74  2 A91 A101   4 A121  61 A143 A152   1 A
172
## 10 A12 30 A34 A40 5234 A61 A71  4 A94 A101   2 A123  28 A143 A152   2 A
174
##    V18  V19  V20 V21
## 1    1 A192 A201   1
## 2    1 A191 A201   2
## 3    2 A191 A201   1
## 4    2 A191 A201   1
## 5    2 A191 A201   2
## 6    2 A192 A201   1
```

```
## 7    1 A191 A201   1
## 8    1 A192 A201   1
## 9    1 A191 A201   1
## 10   1 A191 A201   2
```

2. Data Export

This function prints its required argument x (after converting it to a data frame if it is not one nor a matrix) to a file or connection.

2.1. write.table()

Usage

```
  write.table(x, file = "", append = FALSE, quote = TRUE, sep = " ",
        eol = "\n", na = "NA", dec = ".", row.names = TRUE,
        col.names = TRUE, qmethod = c("escape", "double"),
        fileEncoding = "")
```

```
write.table(mkt.1, "mkt.1.txt", sep = " ")
write.table(mkt.1, "mkt.2.txt", sep = "^")
write.table(mkt.1, "mkt.3.txt", sep = "\t")
```

2.2. write.csv()

Usage

```
  write.csv(...)
```

```
write.csv(ir.1, "iris.csv")
```

3. Functions

Functions are created using the **function()** directive and are stored as R objects. they are R objects of class 'function'

```
f <- function(<arguments>) {
## Do something interesting
}
```

```
myfunction <- function(arg1, arg2, ... ){
  statements
return(object)
}
```

- Functions can be passed as arguments to other functions
- Functions can be nested, so that you can define a function inside of another function
- The return value of a function is the last expression in the function body to be evaluated

3.1. Function Argument

Functions have named *arguments* which potentially have default *values*

- The *formal arguments* are the arguments included in the function definition
- The *formal* function returns a list of all the formal arguments of a function
- Not every function call in R makes use of all the formal arguments
- Function arguments can be *missing* or might have default values

3.2. Argument Matching

R functions arguments can be **matched positionally or by name**. So the following calls to *sd* are all equivalent

```
mydata <- rnorm(100)
sd(mydata)
```

```
## [1] 1.099
```

```
sd(x = mydata)
```

```
## [1] 1.099
```

```
sd(x = mydata, na.rm = FALSE)
```

```
## [1] 1.099
```

```
sd(na.rm = FALSE, x = mydata)
```

```
## [1] 1.099
```

```
sd(na.rm = FALSE, mydata)
```

```
## [1] 1.099
```

```
args(lm)
```

```
## function (formula, data, subset, weights, na.action, method = "qr",
##     model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE,
##     contrasts = NULL, offset, ...)
## NULL
```

You can mix positional matching with matching by name. When an argument is matched by name, it is 'taken out' of the argument list and the remaining unnamed arguments are matched in the order that they are listed in the function definition

```
args(lm)
```

```
## function (formula, data, subset, weights, na.action, method = "qr",
##     model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE,
##     contrasts = NULL, offset, ...)
## NULL
```

The following two calls are equivalent

```
lm(data = cars, dist ~ speed, model = FALSE, 1:100)
```

```
##
## Call:
## lm(formula = dist ~ speed, data = cars, subset = 1:100, model = FALSE)
##
## Coefficients:
## (Intercept)        speed
##      -17.58         3.93
```

```
lm(dist ~ speed, data = cars, 1:100, model = FALSE)
```

```
##
## Call:
## lm(formula = dist ~ speed, data = cars, subset = 1:100, model = FALSE)
##
## Coefficients:
## (Intercept)        speed
##      -17.58         3.93
```

3.3. Defining a Function

```
f <- function(a, b = 1, c = 2, d = NULL) {

}
```

In addition to not specifying a default value, you can also set an argument value to NULL.

3.4. Lazy Evaluation

Arguments to functions are evaluated *lazily*, so they are evaluated only as needed.

```
f.3 <- function(a, b) {
    a^2
}
```

```
f.3(2)
```

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

This function never actually uses the argument b, so calling f(2) will not produce an error because the 2 gets positionally matched to **a**.

Another example

```
f.4 <- function(a, b) {
    print(a)
    print(b)
}
```

```
f.4(45)
```

```
## [1] 45
```

```
## Error: 기본값이 없는 인수 "b"가 누락되어 있습니다
```

Notice that '45' got printed first before the error was triggered. This is because **b** did not have to be evaluated until after **print(a)**. Once the function tried to evaluate **print(b)** it had to throw an error.

3.5 THe '...' Argument

The ... argument indicate a variable number of arguments that are usually passed on to other functions.

- ... is often used when extending another function and you don't want to copy the entire argument list of the original function myplot <- function(x, y, type='l', ...){ plot(x, y, type = type, ...) }
- Generic functions use ... so that extra arguments can be passed to methods (more on this later).

```
mean
```

```
## function (x, ...)
## UseMethod("mean")
## <bytecode: 0x0000000006b33518>
## <environment: namespace:base>
```

The ... argument is also necessary when the number of arguments passed to the function cannot be known in advance.

```
args(paste)
```

```
## function (..., sep = " ", collapse = NULL)
## NULL
```

```
args(cat)
```

```
## function (..., file = "", sep = " ", fill = FALSE, labels = NULL,
##     append = FALSE)
## NULL
```

3.6 Arguments Coming After the '...' Argument

Once catch with ... is that any arguments that appear after ... on the argument list must be named explicitly and cannot be partially matched.

```
args(paste)
```

```
## function (..., sep = " ", collapse = NULL)
## NULL
```

```
paste("a", "b", sep = ":")
```

```
## [1] "a:b"
```

```
paste("a", "b", se = ":")
```

```
## [1] "a b :"
```

4. Example: Multivariate Regression

```
### Multivariate Regression User-Defined Function

MyReg <- function(X) {
    Y <- as.matrix(X[, 1])  # response variable in the first column
    x <- cbind(1, as.matrix(X[, -1]))  # the explanatory variables followi
ng

    n <- nrow(x)  # Number of Obervations
    p <- ncol(x) - 1  # Number of Explanatory variables

    beta <- (solve((t(x)) %*% x)) %*% (t(x)) %*% Y
    pred <- x %*% beta  # Compute Fitted Values
    residuals <- Y - pred  # Compute Residuals
    sig2 <- ((t(residuals)) %*% residuals)/(n - p - 1)  # Compute Variance.

    SST <- sum((Y - mean(Y))^2)  # compute SST
    df.SST <- n - 1
    SSR <- sum((pred - mean(Y))^2)  # compute SSR
    df.SSR <- p
    MSR <- SSR/df.SSR  # compute MSR
    SSE <- sum((Y - pred)^2)  # compute SSE
    df.SSE <- n - p - 1
    MSE <- SSE/df.SSE  # compute MSE

    f.stat <- MSR/MSE  # compute F-statistic
    f.p.value <- pf(f.stat, df.SSR, df.SSE, lower.tail = FALSE)  # compute
 p-value for F-statistic
    R2 <- SSR/SST  # compute R square

    C <- solve((t(x)) %*% x)  # define matrix C
    std.error <- sqrt(sig2 * diag(C))  # compute Standard error for parame
te estimates

    t <- beta/std.error  # compute t-statistic
    t.p.value <- round(pt(abs(t), df.SSE, lower.tail = FALSE) + pt(-abs(t),

        df.SSE), 4)
    # compute p-value for t-statistic

    H <- x %*% (solve((t(x)) %*% x)) %*% (t(x))  # hat matrix
    std.resid <- residuals/(sqrt(sig2 * (1 - diag(H))))  # Studentized res
iduals

    # Result List

    res <- list(beta = as.vector(beta), sig2 = sig2, pred = pred, residual
s = residuals,
        stdresid = std.resid, SSR = SSR, SSE = SSE, F = f.stat, P.value =
```

```
f.p.value,
        dat = X)

    # ANOVA table asterisks(*) defined function
    ast <- function(d) {
        if (d >= 0.05 && d < 0.1)
            "." else if (d >= 0.01 && d < 0.05)
            "*" else if (d >= 0.001 && d < 0.01)
            "**" else if (d < 0.001)
            "***" else " "
    }

    cat("\n  == ANALYSIS OF VARIANCE ==\n\n", encodeString(c("    Source",
 "df",
        "SS", "MS", "F", "P-value"), width = 8, justify = "r"), "\n", "---
----------------------------------------------------\n",
        encodeString(c("Regression", df.SSR, round(SSR, 2), round(MSR, 2),
 round(f.stat,
            2), round(f.p.value, 2)), width = 8, justify = "r"), ast(f.p.v
alue),
        "\n", encodeString(c("     Error", df.SSE, round(SSE, 2), round(MS
E,
            2)), width = 8, justify = "r"), "\n", encodeString(c("     Tot
al",
            df.SST, round(SST, 2)), width = 8, justify = "r"), "\n", "----
----------------------------------------------------\n",
        paste("Estimated error variance :", round(sig2, 4)), "\n", paste("
R-squares :",
            round(R2, 4)), "\n\n\n")
    # ANOVA End



    # Prameter estimates all variables name - 1st column
    pe.name <- colnames(X)
    pe.name[1] <- "(Intercept)"

    # variable name + beta + std.error + t value + Pr(>|t|) + asterisks(*)
    pe <- matrix(c(pe.name, round(beta, 4), round(std.error, 4), round(t,
4),
        round(t.p.value, 4)), nr = length(pe.name))

    cat("  == PARAMETER ESTIMATES ==\n\n", encodeString(c("           ", "
Estimate",
        "Std.Error", "t value", "Pr(>|t|)"), width = 11, justify = "r"), "
\n")

    for (i in 1:length(pe.name)) {
        cat(encodeString(pe[i, ], width = 11, justify = "r"), ast(pe[i,
5]),
            "\n")
```

```
    }
    return(res)
}

MyReg(cars)  # simple linear regression
```

```
##
##   == ANALYSIS OF VARIANCE ==
##
##      Source       df       SS       MS        F  P-value
##  ------------------------------------------------------
##  Regression        1   891.98   891.98    89.57        0 ***
##       Error       48   478.02     9.96
##       Total       49     1370
##  ------------------------------------------------------
##  Estimated error variance : 9.9588
##  R-squares : 0.6511
##
##
##   == PARAMETER ESTIMATES ==
##
##                Estimate   Std.Error     t value    Pr(>|t|)
## (Intercept)      8.2839      0.8744       9.474           0 ***
##        dist      0.1656      0.0175       9.464           0 ***

## $beta
## [1] 8.2839 0.1656
##
## $sig2
##       [,1]
## [1,] 9.959
##
## $pred
##         [,1]
##  [1,]  8.615
##  [2,]  9.940
##  [3,]  8.946
##  [4,] 11.926
##  [5,] 10.933
##  [6,]  9.940
##  [7,] 11.264
##  [8,] 12.589
##  [9,] 13.913
## [10,] 11.099
## [11,] 12.920
## [12,] 10.602
## [13,] 11.595
## [14,] 12.258
## [15,] 12.920
## [16,] 12.589
## [17,] 13.913
## [18,] 13.913
## [19,] 15.900
```

```
## [20,] 12.589
## [21,] 14.244
## [22,] 18.218
## [23,] 21.529
## [24,] 11.595
## [25,] 12.589
## [26,] 17.225
## [27,] 13.582
## [28,] 14.907
## [29,] 13.582
## [30,] 14.907
## [31,] 16.562
## [32,] 15.238
## [33,] 17.556
## [34,] 20.867
## [35,] 22.192
## [36,] 14.244
## [37,] 15.900
## [38,] 19.543
## [39,] 13.582
## [40,] 16.231
## [41,] 16.893
## [42,] 17.556
## [43,] 18.880
## [44,] 19.211
## [45,] 17.225
## [46,] 19.874
## [47,] 23.516
## [48,] 23.682
## [49,] 28.152
## [50,] 22.357
##
## $residuals
##           [,1]
##  [1,] -4.61504
##  [2,] -5.93958
##  [3,] -1.94618
##  [4,] -4.92639
##  [5,] -2.93299
##  [6,] -0.93958
##  [7,] -1.26412
##  [8,] -2.58866
##  [9,] -3.91320
## [10,] -0.09855
## [11,] -1.91980
## [12,]  1.39815
## [13,]  0.40474
## [14,] -0.25753
## [15,] -0.91980
## [16,]  0.41134
## [17,] -0.91320
## [18,] -0.91320
## [19,] -2.90001
```

```
## [20,]  1.41134
## [21,] -0.24434
## [22,] -4.21796
## [23,] -7.52931
## [24,]  3.40474
## [25,]  2.41134
## [26,] -2.22455
## [27,]  2.41793
## [28,]  1.09339
## [29,]  3.41793
## [30,]  2.09339
## [31,]  0.43772
## [32,]  2.76226
## [33,]  0.44431
## [34,] -2.86704
## [35,] -4.19158
## [36,]  4.75566
## [37,]  3.09999
## [38,] -0.54250
## [39,]  6.41793
## [40,]  3.76885
## [41,]  3.10658
## [42,]  2.44431
## [43,]  1.11977
## [44,]  2.78863
## [45,]  5.77545
## [46,]  4.12636
## [47,]  0.48388
## [48,]  0.31831
## [49,] -4.15201
## [50,]  2.64285
##
## $stdresid
##           [,1]
##  [1,] -1.51778
##  [2,] -1.93453
##  [3,] -0.63836
##  [4,] -1.58793
##  [5,] -0.94975
##  [6,] -0.30602
##  [7,] -0.40866
##  [8,] -0.83240
##  [9,] -1.25420
## [10,] -0.03189
## [11,] -0.61670
## [12,]  0.45356
## [13,]  0.13064
## [14,] -0.08290
## [15,] -0.29547
## [16,]  0.13227
## [17,] -0.29269
## [18,] -0.29269
## [19,] -0.92842
```

```
## [20,]  0.45382
## [21,] -0.07827
## [22,] -1.35634
## [23,] -2.46365
## [24,]  1.09899
## [25,]  0.77538
## [26,] -0.71344
## [27,]  0.77544
## [28,]  0.35004
## [29,]  1.09615
## [30,]  0.67019
## [31,]  0.14022
## [32,]  0.88421
## [33,]  0.14260
## [34,] -0.93384
## [35,] -1.37858
## [36,]  1.52345
## [37,]  0.99244
## [38,] -0.17538
## [39,]  2.05827
## [40,]  1.20688
## [41,]  0.99568
## [42,]  0.78451
## [43,]  0.36095
## [44,]  0.90015
## [45,]  1.85224
## [46,]  1.33623
## [47,]  0.16108
## [48,]  0.10614
## [49,] -1.47312
## [50,]  0.87041
##
## $SSR
## [1] 892
##
## $SSE
## [1] 478
##
## $F
## [1] 89.57
##
## $P.value
## [1] 1.49e-12
##
## $dat
##    speed dist
## 1      4    2
## 2      4   10
## 3      7    4
## 4      7   22
## 5      8   16
## 6      9   10
## 7     10   18
```

```
## 8     10   26
## 9     10   34
## 10    11   17
## 11    11   28
## 12    12   14
## 13    12   20
## 14    12   24
## 15    12   28
## 16    13   26
## 17    13   34
## 18    13   34
## 19    13   46
## 20    14   26
## 21    14   36
## 22    14   60
## 23    14   80
## 24    15   20
## 25    15   26
## 26    15   54
## 27    16   32
## 28    16   40
## 29    17   32
## 30    17   40
## 31    17   50
## 32    18   42
## 33    18   56
## 34    18   76
## 35    18   84
## 36    19   36
## 37    19   46
## 38    19   68
## 39    20   32
## 40    20   48
## 41    20   52
## 42    20   56
## 43    20   64
## 44    22   66
## 45    23   54
## 46    24   70
## 47    24   92
## 48    24   93
## 49    24  120
## 50    25   85
```

```
MyReg(mtcars)  # multivariate regression
```

```
##
##   == ANALYSIS OF VARIANCE ==
##
##      Source       df       SS       MS        F  P-value
##  ------------------------------------------------------
##  Regression       10   978.55    97.86    13.93        0 ***
```

```
##       Error       21   147.49     7.02
##       Total       31  1126.05
##  ----------------------------------------------------
##  Estimated error variance : 7.0235
##  R-squares : 0.869
##
##
##   == PARAMETER ESTIMATES ==
##
##                 Estimate   Std.Error     t value    Pr(>|t|)
## (Intercept)     12.3034     18.7179      0.6573      0.5181
##         cyl     -0.1114       1.045     -0.1066      0.9161
##        disp      0.0133      0.0179      0.7468      0.4635
##          hp     -0.0215      0.0218     -0.9868       0.335
##        drat      0.7871      1.6354      0.4813      0.6353
##          wt     -3.7153      1.8944     -1.9612      0.0633 .
##        qsec       0.821      0.7308      1.1234      0.2739
##          vs      0.3178      2.1045       0.151      0.8814
##          am      2.5202      2.0567      1.2254       0.234
##        gear      0.6554      1.4933      0.4389      0.6652
##        carb     -0.1994      0.8288     -0.2406      0.8122

## $beta
##  [1] 12.30337 -0.11144  0.01334 -0.02148  0.78711 -3.71530  0.82104
##  [8]  0.31776  2.52023  0.65541 -0.19942
##
## $sig2
##       [,1]
## [1,] 7.024
##
## $pred
##                      [,1]
## Mazda RX4           22.60
## Mazda RX4 Wag       22.11
## Datsun 710          26.25
## Hornet 4 Drive      21.24
## Hornet Sportabout   17.69
## Valiant             20.38
## Duster 360          14.39
## Merc 240D           22.50
## Merc 230            24.42
## Merc 280            18.70
## Merc 280C           19.19
## Merc 450SE          14.17
## Merc 450SL          15.60
## Merc 450SLC         15.74
## Cadillac Fleetwood  12.03
## Lincoln Continental 10.94
## Chrysler Imperial   10.49
## Fiat 128            27.77
## Honda Civic         29.90
## Toyota Corolla      29.51
## Toyota Corona       23.64
```

```
## Dodge Challenger    16.94
## AMC Javelin         17.73
## Camaro Z28          13.31
## Pontiac Firebird    16.69
## Fiat X1-9           28.29
## Porsche 914-2       26.15
## Lotus Europa        27.64
## Ford Pantera L      18.87
## Ferrari Dino        19.69
## Maserati Bora       13.94
## Volvo 142E          24.37
##
## $residuals
##                          [,1]
## Mazda RX4           -1.599506
## Mazda RX4 Wag       -1.111886
## Datsun 710          -3.450644
## Hornet 4 Drive       0.162595
## Hornet Sportabout    1.006566
## Valiant             -2.283039
## Duster 360          -0.086256
## Merc 240D            1.903988
## Merc 230            -1.619090
## Merc 280             0.500970
## Merc 280C           -1.391654
## Merc 450SE           2.227838
## Merc 450SL           1.700426
## Merc 450SLC         -0.542225
## Cadillac Fleetwood  -1.634013
## Lincoln Continental -0.536438
## Chrysler Imperial    4.206371
## Fiat 128             4.627094
## Honda Civic          0.503261
## Toyota Corolla       4.387631
## Toyota Corona       -2.143103
## Dodge Challenger    -1.443053
## AMC Javelin         -2.532181
## Camaro Z28          -0.006022
## Pontiac Firebird     2.508321
## Fiat X1-9           -0.993469
## Porsche 914-2       -0.152954
## Lotus Europa         2.763727
## Ford Pantera L      -3.070041
## Ferrari Dino         0.006172
## Maserati Bora        1.058882
## Volvo 142E          -2.968268
##
## $stdresid
##                          [,1]
## Mazda RX4           -0.722666
## Mazda RX4 Wag       -0.497990
## Datsun 710          -1.492373
## Hornet 4 Drive       0.069815
```

```
## Hornet Sportabout    0.424509
## Valiant             -1.016855
## Duster 360          -0.039642
## Merc 240D            0.877857
## Merc 230            -1.203440
## Merc 280             0.250230
## Merc 280C           -0.664146
## Merc 450SE           1.007109
## Merc 450SL           0.713847
## Merc 450SLC         -0.232207
## Cadillac Fleetwood  -0.779573
## Lincoln Continental -0.243510
## Chrysler Imperial    1.906196
## Fiat 128             1.926840
## Honda Civic          0.271816
## Toyota Corolla       1.890244
## Toyota Corona       -1.074316
## Dodge Challenger    -0.615749
## AMC Javelin         -1.051583
## Camaro Z28          -0.002953
## Pontiac Firebird     1.061707
## Fiat X1-9           -0.404738
## Porsche 914-2       -0.094025
## Lotus Europa         1.382606
## Ford Pantera L      -1.996242
## Ferrari Dino         0.002984
## Maserati Bora        0.668479
## Volvo 142E          -1.329692
##
## $SSR
## [1] 978.6
##
## $SSE
## [1] 147.5
##
## $F
## [1] 13.93
##
## $P.value
## [1] 3.793e-07
##
## $dat
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
```

```
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

Reference

- [Introduction to the R Language] (http://www.stat.berkeley.edu/~statcur/Workshop2/Presentations/functions.pdf)