Laporan Praktikum 1 AMP

Antonius Aditya Rizky Wijaya

G5402221003

2025-01-23

setwd("//sapi/HOMEMHS/AKT2022/g5402221003/Analisis Model Prediktif")

## Basic Commands

x <- c(1, 3, 2, 5)  
x

## [1] 1 3 2 5

Note that the > is not part of the command; rather, it is printed by R to indicate that it is ready for another command to be entered. We can also save things using = rather than <-:

x = c(1, 6, 2)  
x

## [1] 1 6 2

y = c(1, 4, 3)

Hitting the *up* arrow multiple times will display the previous commands, which can then be edited. This is useful since one often wishes to repeat a similar command. In addition, typing ?funcname will always cause R to open a new help file window with additional information about the function funcname().

We can tell R to add two sets of numbers together. It will then add the first number from x to the first number from y, and so on. However, x and y should be the same length. We can check their length using the length() function.

length(x)

## [1] 3

length(y)

## [1] 3

x + y

## [1] 2 10 5

The ls() function allows us to look at a list of all of the objects, such as data and functions, that we have saved so far. The rm() function can be used to delete any that we don’t want.

ls()

## [1] "x" "y"

rm(x, y)  
ls()

## character(0)

It’s also possible to remove all objects at once:

rm(list = ls())

The matrix() function can be used to create a matrix of numbers. Before we use the matrix() function, we can learn more about it:

?matrix

## starting httpd help server ... done

The help file reveals that the matrix() function takes a number of inputs, but for now we focus on the first three: the data (the entries in the matrix), the number of rows, and the number of columns. First, we create a simple matrix.

x <- matrix(data = c(1, 2, 3, 4), nrow = 2, ncol = 2)  
x

## [,1] [,2]  
## [1,] 1 3  
## [2,] 2 4

Note that we could just as well omit typing data=, nrow=, and ncol= in the matrix() command above: that is, we could just type

x <- matrix(c(1, 2, 3, 4), 2, 2)

and this would have the same effect. However, it can sometimes be useful to specify the names of the arguments passed in, since otherwise R will assume that the function arguments are passed into the function in the same order that is given in the function’s help file. As this example illustrates, by default R creates matrices by successively filling in columns. Alternatively, the byrow = TRUE option can be used to populate the matrix in order of the rows.

matrix(c(1, 2, 3, 4), 2, 2, byrow = TRUE)

## [,1] [,2]  
## [1,] 1 2  
## [2,] 3 4

Notice that in the above command we did not assign the matrix to a value such as x. In this case the matrix is printed to the screen but is not saved for future calculations. The sqrt() function returns the square root of each element of a vector or matrix. The command x^2 raises each element of x to the power 2; any powers are possible, including fractional or negative powers.

sqrt(x)

## [,1] [,2]  
## [1,] 1.000000 1.732051  
## [2,] 1.414214 2.000000

x^2

## [,1] [,2]  
## [1,] 1 9  
## [2,] 4 16

The rnorm() function generates a vector of random normal variables, with first argument n the sample size. Each time we call this function, we will get a different answer. Here we create two correlated sets of numbers, x and y, and use the cor() function to compute the correlation between them.

x <- rnorm(50)  
y <- x + rnorm(50, mean = 50, sd = .1)  
cor(x, y)

## [1] 0.996153

By default, rnorm() creates standard normal random variables with a mean of and a standard deviation of . However, the mean and standard deviation can be altered using the mean and sd arguments, as illustrated above. Sometimes we want our code to reproduce the exact same set of random numbers; we can use the set.seed() function to do this. The set.seed() function takes an (arbitrary) integer argument.

set.seed(1303)  
rnorm(50)

## [1] -1.1439763145 1.3421293656 2.1853904757 0.5363925179 0.0631929665  
## [6] 0.5022344825 -0.0004167247 0.5658198405 -0.5725226890 -1.1102250073  
## [11] -0.0486871234 -0.6956562176 0.8289174803 0.2066528551 -0.2356745091  
## [16] -0.5563104914 -0.3647543571 0.8623550343 -0.6307715354 0.3136021252  
## [21] -0.9314953177 0.8238676185 0.5233707021 0.7069214120 0.4202043256  
## [26] -0.2690521547 -1.5103172999 -0.6902124766 -0.1434719524 -1.0135274099  
## [31] 1.5732737361 0.0127465055 0.8726470499 0.4220661905 -0.0188157917  
## [36] 2.6157489689 -0.6931401748 -0.2663217810 -0.7206364412 1.3677342065  
## [41] 0.2640073322 0.6321868074 -1.3306509858 0.0268888182 1.0406363208  
## [46] 1.3120237985 -0.0300020767 -0.2500257125 0.0234144857 1.6598706557

We use set.seed() throughout the labs whenever we perform calculations involving random quantities. In general this should allow the user to reproduce our results. However, as new versions of R become available, small discrepancies may arise between this book and the output from R.

The mean() and var() functions can be used to compute the mean and variance of a vector of numbers. Applying sqrt() to the output of var() will give the standard deviation. Or we can simply use the sd() function.

set.seed(3)  
y <- rnorm(100)  
mean(y)

## [1] 0.01103557

var(y)

## [1] 0.7328675

sqrt(var(y))

## [1] 0.8560768

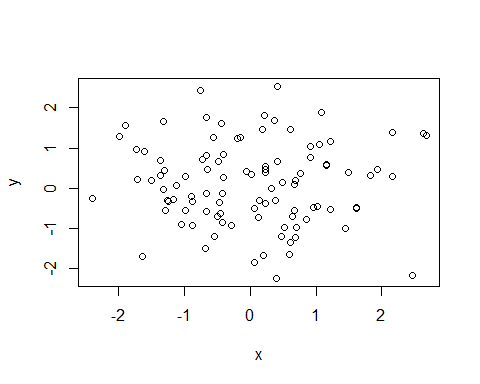
sd(y)

## [1] 0.8560768

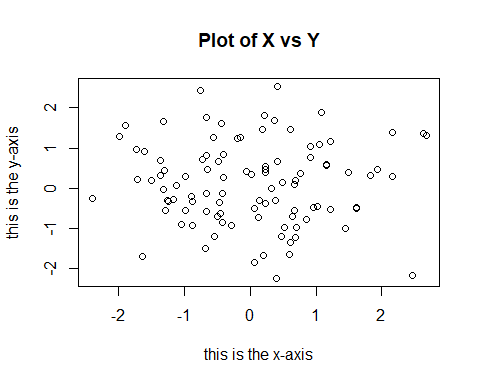
## Graphics

The plot() function is the primary way to plot data in R. For instance, plot(x, y) produces a scatterplot of the numbers in x versus the numbers in y. There are many additional options that can be passed in to the plot() function.

x <- rnorm(100)  
y <- rnorm(100)  
plot(x, y)



plot(x, y, xlab = "this is the x-axis",  
 ylab = "this is the y-axis",  
 main = "Plot of X vs Y")



We will often want to save the output of an R plot. The command that we use to do this will depend on the file type that we would like to create. For instance, to create a pdf, we use the pdf() function, and to create a jpeg, we use the jpeg() function.

pdf("Figure.pdf")  
plot(x, y, col = "green")  
dev.off()

## png   
## 2

The function dev.off() indicates to R that we are done creating the plot. Alternatively, we can simply copy the plot window and paste it into an appropriate file type, such as a Word document.

The function seq() can be used to create a sequence of numbers. For instance, seq(a, b) makes a vector of integers between a and b. There are many other options: for instance, seq(0, 1, length = 10) makes a sequence of 10 numbers that are equally spaced between 0 and 1. Typing 3:11 is a shorthand for seq(3, 11) for integer arguments.

x <- seq(1, 10)  
x

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

x <- 1:10  
x

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

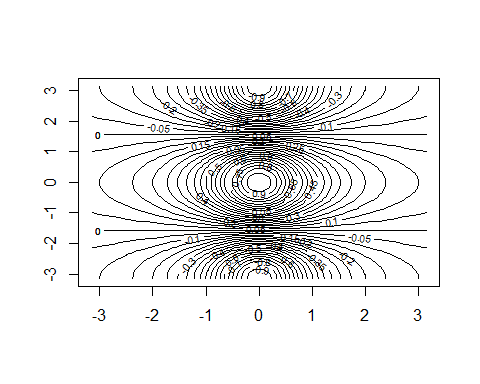
x <- seq(-pi, pi, length = 50)

We will now create some more sophisticated plots. The contour() function produces a *contour plot* in order to represent three-dimensional data; it is like a topographical map. It takes three arguments:

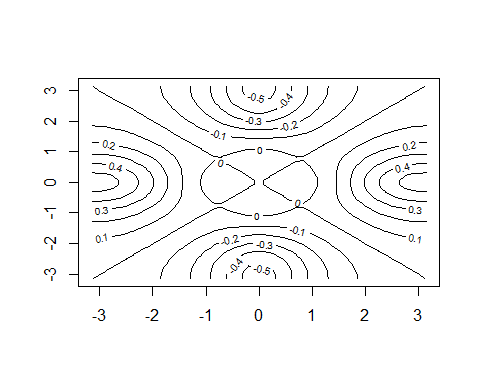
* A vector of the x values (the first dimension),
* A vector of the y values (the second dimension), and
* A matrix whose elements correspond to the z value (the third dimension) for each pair of (x, y) coordinates.

As with the plot() function, there are many other inputs that can be used to fine-tune the output of the contour() function. To learn more about these, take a look at the help file by typing ?contour.

y <- x  
f <- outer(x, y, function(x, y) cos(y) / (1 + x^2))  
contour(x, y, f)  
contour(x, y, f, nlevels = 45, add = T)

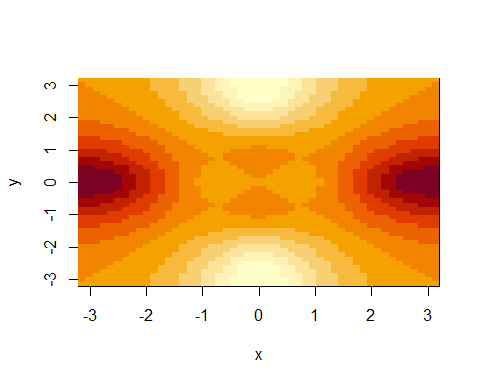


fa <- (f - t(f)) / 2  
contour(x, y, fa, nlevels = 15)

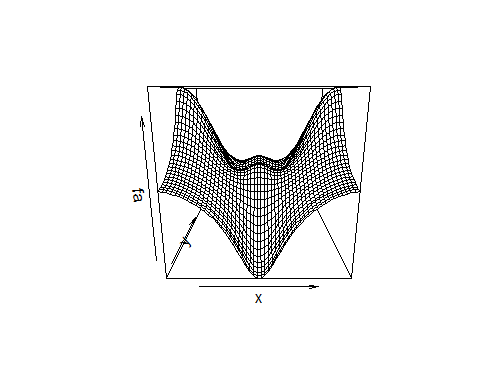


The image() function works the same way as contour(), except that it produces a color-coded plot whose colors depend on the z value. This is known as a *heatmap*, and is sometimes used to plot temperature in weather forecasts. Alternatively, persp() can be used to produce a three-dimensional plot. The arguments theta and phi control the angles at which the plot is viewed.

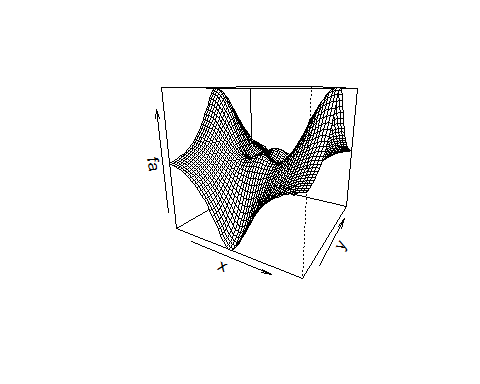
image(x, y, fa)



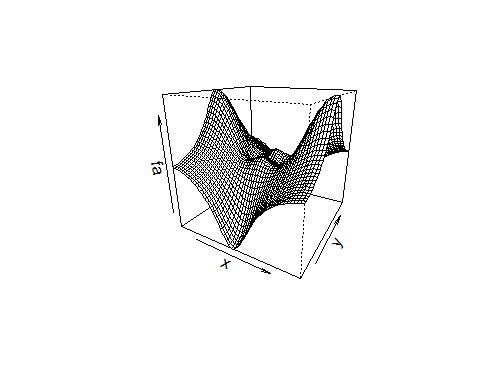
persp(x, y, fa)



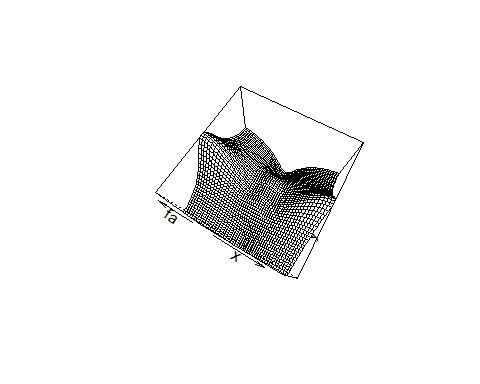
persp(x, y, fa, theta = 30)



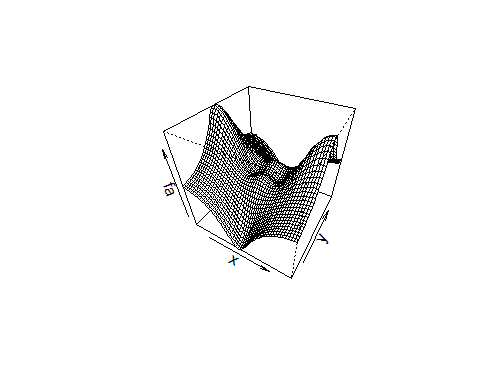
persp(x, y, fa, theta = 30, phi = 20)



persp(x, y, fa, theta = 30, phi = 70)



persp(x, y, fa, theta = 30, phi = 40)



## Indexing Data

We often wish to examine part of a set of data. Suppose that our data is stored in the matrix A.

A <- matrix(1:16, 4, 4)  
A

## [,1] [,2] [,3] [,4]  
## [1,] 1 5 9 13  
## [2,] 2 6 10 14  
## [3,] 3 7 11 15  
## [4,] 4 8 12 16

Then, typing

A[2, 3]

## [1] 10

will select the element corresponding to the second row and the third column. The first number after the open-bracket symbol [ always refers to the row, and the second number always refers to the column. We can also select multiple rows and columns at a time, by providing vectors as the indices.

A[c(1, 3), c(2, 4)]

## [,1] [,2]  
## [1,] 5 13  
## [2,] 7 15

A[1:3, 2:4]

## [,1] [,2] [,3]  
## [1,] 5 9 13  
## [2,] 6 10 14  
## [3,] 7 11 15

A[1:2, ]

## [,1] [,2] [,3] [,4]  
## [1,] 1 5 9 13  
## [2,] 2 6 10 14

A[, 1:2]

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

The last two examples include either no index for the columns or no index for the rows. These indicate that R should include all columns or all rows, respectively. R treats a single row or column of a matrix as a vector.

A[1, ]

## [1] 1 5 9 13

The use of a negative sign - in the index tells R to keep all rows or columns except those indicated in the index.

A[-c(1, 3), ]

## [,1] [,2] [,3] [,4]  
## [1,] 2 6 10 14  
## [2,] 4 8 12 16

A[-c(1, 3), -c(1, 3, 4)]

## [1] 6 8

The dim() function outputs the number of rows followed by the number of columns of a given matrix.

dim(A)

## [1] 4 4

## Loading Data

#Auto <- read.table("Auto.data")  
#View(Auto)  
#head(Auto)

Note that Auto.data is simply a text file, which you could alternatively open on your computer using a standard text editor. It is often a good idea to view a data set using a text editor or other software such as Excel before loading it into R.

This particular data set has not been loaded correctly, because R has assumed that the variable names are part of the data and so has included them in the first row. The data set also includes a number of missing observations, indicated by a question mark ?. Missing values are a common occurrence in real data sets. Using the option header = T (or header = TRUE) in the read.table() function tells R that the first line of the file contains the variable names, and using the option na.strings tells R that any time it sees a particular character or set of characters (such as a question mark), it should be treated as a missing element of the data matrix.

#Auto <- read.table("Auto.data", header = T, na.strings = "?", stringsAsFactors = T)  
#View(Auto)

The stringsAsFactors = T argument tells R that any variable containing character strings should be interpreted as a qualitative variable, and that each distinct character string represents a distinct level for that qualitative variable. An easy way to load data from Excel into R is to save it as a csv (comma-separated values) file, and then use the read.csv() function.

Auto <- read.csv("Auto.csv", na.strings = "?", stringsAsFactors = T)  
View(Auto)  
dim(Auto)

## [1] 397 9

Auto[1:4, ]

## mpg cylinders displacement horsepower weight acceleration year origin  
## 1 18 8 307 130 3504 12.0 70 1  
## 2 15 8 350 165 3693 11.5 70 1  
## 3 18 8 318 150 3436 11.0 70 1  
## 4 16 8 304 150 3433 12.0 70 1  
## name  
## 1 chevrolet chevelle malibu  
## 2 buick skylark 320  
## 3 plymouth satellite  
## 4 amc rebel sst

The dim() function tells us that the data has observations, or rows, and nine variables, or columns. There are various ways to deal with the missing data. In this case, only five of the rows contain missing observations, and so we choose to use the na.omit() function to simply remove these rows.

Auto <- na.omit(Auto)  
dim(Auto)

## [1] 392 9

Once the data are loaded correctly, we can use names() to check the variable names.

names(Auto)

## [1] "mpg" "cylinders" "displacement" "horsepower" "weight"   
## [6] "acceleration" "year" "origin" "name"

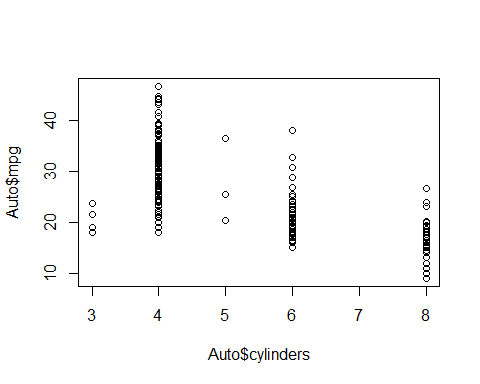
## Additional Graphical and Numerical Summaries

We can use the plot() function to produce *scatterplots* of the quantitative variables. However, simply typing the variable names will produce an error message, because R does not know to look in the Auto data set for those variables.

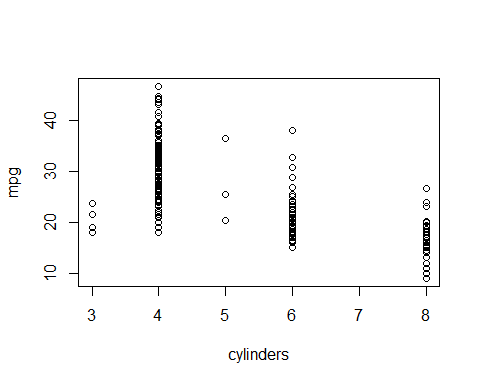
#plot(cylinders, mpg)

To refer to a variable, we must type the data set and the variable name joined with a $ symbol. Alternatively, we can use the attach() function in order to tell R to make the variables in this data frame available by name.

plot(Auto$cylinders, Auto$mpg)



attach(Auto)  
plot(cylinders, mpg)

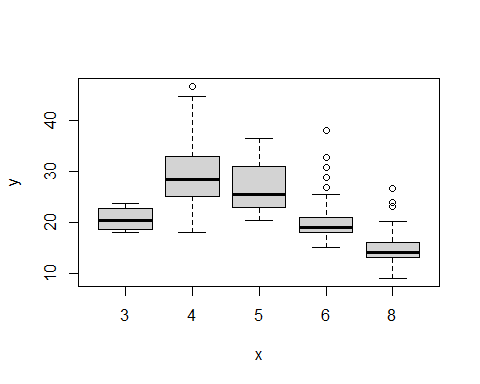


The cylinders variable is stored as a numeric vector, so R has treated it as quantitative. However, since there are only a small number of possible values for cylinders, one may prefer to treat it as a qualitative variable. The as.factor() function converts quantitative variables into qualitative variables.

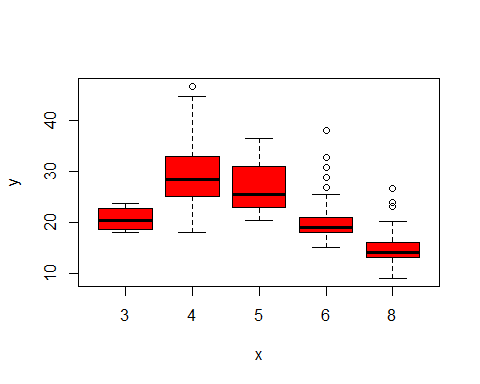
cylinders <- as.factor(cylinders)

If the variable plotted on the -axis is qualitative, then *boxplots* will automatically be produced by the plot() function. As usual, a number of options can be specified in order to customize the plots.

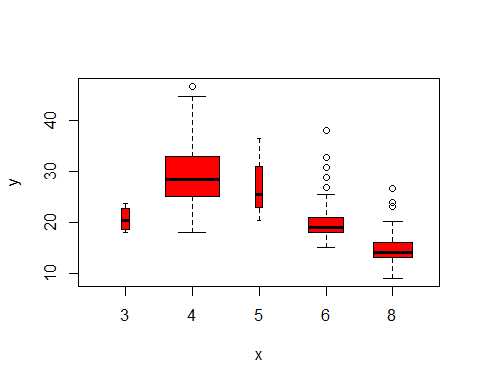
plot(cylinders, mpg)



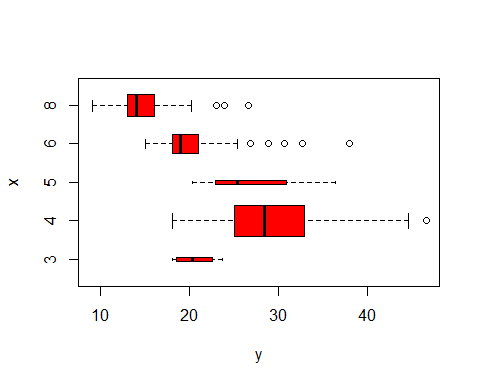
plot(cylinders, mpg, col = "red")



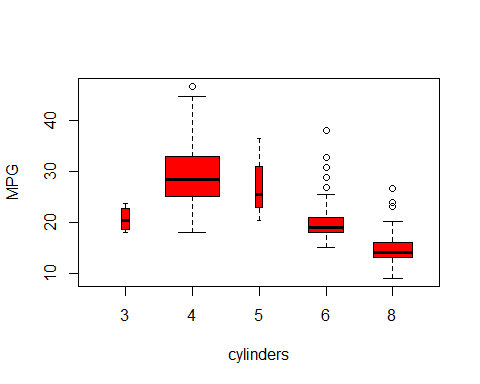
plot(cylinders, mpg, col = "red", varwidth = T)



plot(cylinders, mpg, col = "red", varwidth = T,  
 horizontal = T)

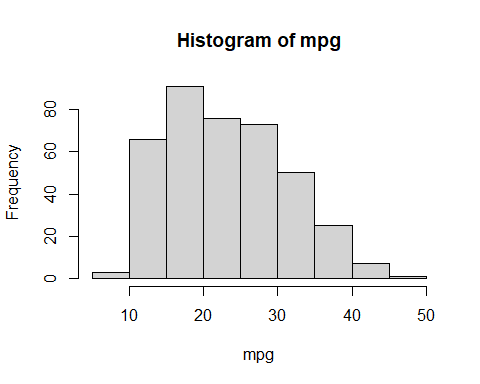


plot(cylinders, mpg, col = "red", varwidth = T,  
 xlab = "cylinders", ylab = "MPG")

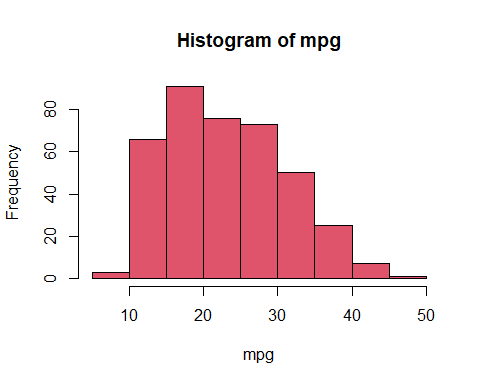


The hist() function can be used to plot a *histogram*. Note that col = 2 has the same effect as col = "red".

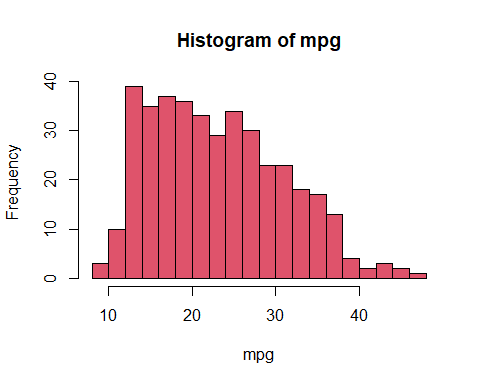
hist(mpg)



hist(mpg, col = 2)

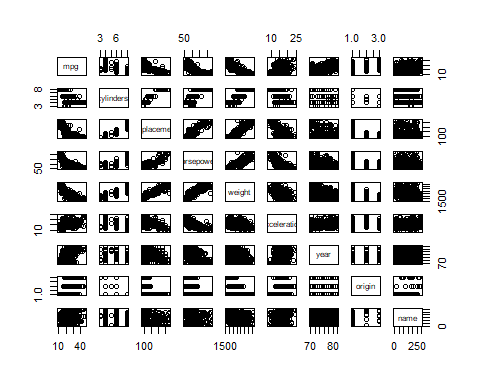


hist(mpg, col = 2, breaks = 15)

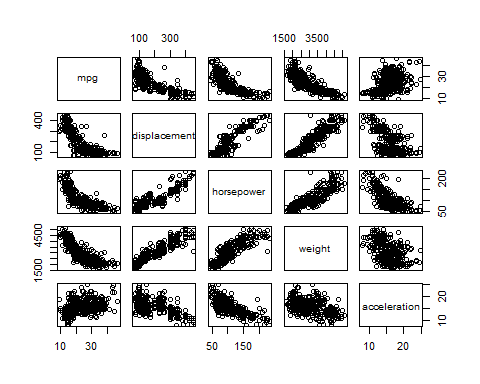


The pairs() function creates a *scatterplot matrix*, i.e. a scatterplot for every pair of variables. We can also produce scatterplots for just a subset of the variables.

pairs(Auto)

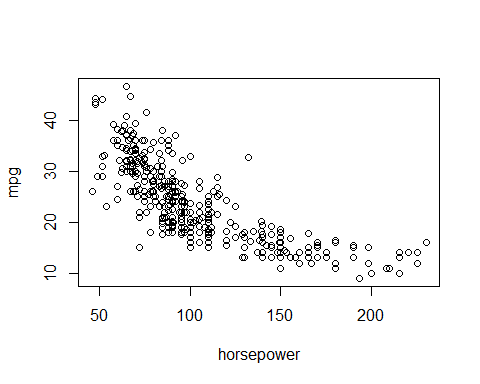


pairs(  
 ~ mpg + displacement + horsepower + weight + acceleration,  
 data = Auto  
 )



In conjunction with the plot() function, identify() provides a useful interactive method for identifying the value of a particular variable for points on a plot. We pass in three arguments to identify(): the -axis variable, the -axis variable, and the variable whose values we would like to see printed for each point. Then clicking one or more points in the plot and hitting Escape will cause R to print the values of the variable of interest. The numbers printed under the identify() function correspond to the rows for the selected points.

plot(horsepower, mpg)  
identify(horsepower, mpg, name)



## integer(0)

The summary() function produces a numerical summary of each variable in a particular data set.

summary(Auto)

## mpg cylinders displacement horsepower weight   
## Min. : 9.00 Min. :3.000 Min. : 68.0 Min. : 46.0 Min. :1613   
## 1st Qu.:17.00 1st Qu.:4.000 1st Qu.:105.0 1st Qu.: 75.0 1st Qu.:2225   
## Median :22.75 Median :4.000 Median :151.0 Median : 93.5 Median :2804   
## Mean :23.45 Mean :5.472 Mean :194.4 Mean :104.5 Mean :2978   
## 3rd Qu.:29.00 3rd Qu.:8.000 3rd Qu.:275.8 3rd Qu.:126.0 3rd Qu.:3615   
## Max. :46.60 Max. :8.000 Max. :455.0 Max. :230.0 Max. :5140   
##   
## acceleration year origin name   
## Min. : 8.00 Min. :70.00 Min. :1.000 amc matador : 5   
## 1st Qu.:13.78 1st Qu.:73.00 1st Qu.:1.000 ford pinto : 5   
## Median :15.50 Median :76.00 Median :1.000 toyota corolla : 5   
## Mean :15.54 Mean :75.98 Mean :1.577 amc gremlin : 4   
## 3rd Qu.:17.02 3rd Qu.:79.00 3rd Qu.:2.000 amc hornet : 4   
## Max. :24.80 Max. :82.00 Max. :3.000 chevrolet chevette: 4   
## (Other) :365

For qualitative variables such as name, R will list the number of observations that fall in each category. We can also produce a summary of just a single variable.

summary(mpg)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 9.00 17.00 22.75 23.45 29.00 46.60

### 

### NOMOR 8

1. Use the read.csv() function to read the data into R. Call the loaded data college. Make sure that you have the directory set to the correct location for the data.

college <- read.csv("College.csv")

1. Look at the data using the View() function. You should notice that the first column is just the name of each university. We don’t really want R to treat this as data. However, it may be handy to have these names for later.

rownames(college) <- college[, 1]  
View(college)  
  
college <- college[, -1]  
View(college)

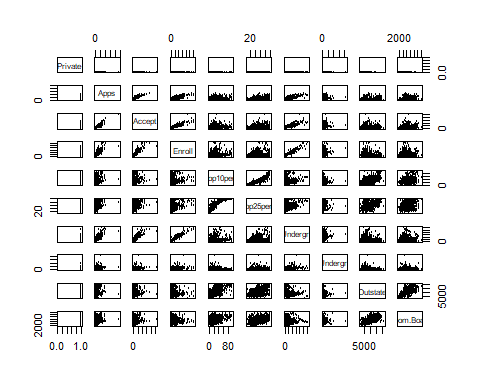
* 1. Use the summary() function to produce a numerical summary of the variables in the data set.

summary(college)

## Private Apps Accept Enroll   
## Length:777 Min. : 81 Min. : 72 Min. : 35   
## Class :character 1st Qu.: 776 1st Qu.: 604 1st Qu.: 242   
## Mode :character Median : 1558 Median : 1110 Median : 434   
## Mean : 3002 Mean : 2019 Mean : 780   
## 3rd Qu.: 3624 3rd Qu.: 2424 3rd Qu.: 902   
## Max. :48094 Max. :26330 Max. :6392   
## Top10perc Top25perc F.Undergrad P.Undergrad   
## Min. : 1.00 Min. : 9.0 Min. : 139 Min. : 1.0   
## 1st Qu.:15.00 1st Qu.: 41.0 1st Qu.: 992 1st Qu.: 95.0   
## Median :23.00 Median : 54.0 Median : 1707 Median : 353.0   
## Mean :27.56 Mean : 55.8 Mean : 3700 Mean : 855.3   
## 3rd Qu.:35.00 3rd Qu.: 69.0 3rd Qu.: 4005 3rd Qu.: 967.0   
## Max. :96.00 Max. :100.0 Max. :31643 Max. :21836.0   
## Outstate Room.Board Books Personal   
## Min. : 2340 Min. :1780 Min. : 96.0 Min. : 250   
## 1st Qu.: 7320 1st Qu.:3597 1st Qu.: 470.0 1st Qu.: 850   
## Median : 9990 Median :4200 Median : 500.0 Median :1200   
## Mean :10441 Mean :4358 Mean : 549.4 Mean :1341   
## 3rd Qu.:12925 3rd Qu.:5050 3rd Qu.: 600.0 3rd Qu.:1700   
## Max. :21700 Max. :8124 Max. :2340.0 Max. :6800   
## PhD Terminal S.F.Ratio perc.alumni   
## Min. : 8.00 Min. : 24.0 Min. : 2.50 Min. : 0.00   
## 1st Qu.: 62.00 1st Qu.: 71.0 1st Qu.:11.50 1st Qu.:13.00   
## Median : 75.00 Median : 82.0 Median :13.60 Median :21.00   
## Mean : 72.66 Mean : 79.7 Mean :14.09 Mean :22.74   
## 3rd Qu.: 85.00 3rd Qu.: 92.0 3rd Qu.:16.50 3rd Qu.:31.00   
## Max. :103.00 Max. :100.0 Max. :39.80 Max. :64.00   
## Expend Grad.Rate   
## Min. : 3186 Min. : 10.00   
## 1st Qu.: 6751 1st Qu.: 53.00   
## Median : 8377 Median : 65.00   
## Mean : 9660 Mean : 65.46   
## 3rd Qu.:10830 3rd Qu.: 78.00   
## Max. :56233 Max. :118.00

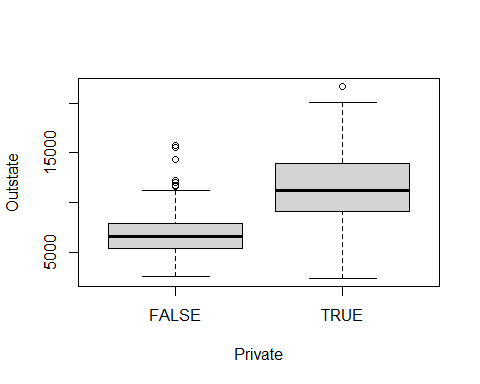
ii. Use the `pairs()` function to produce a scatterplot matrix of the  
 first ten columns or variables of the data. Recall that you can   
 reference the first ten columns of a matrix A using `A[,1:10]`.

college$Private <- college$Private == "Yes"  
pairs(college[, 1:10], cex = 0.2)



iii. Use the `plot()` function to produce side-by-side boxplots of   
 `Outstate` versus `Private`.

plot(college$Outstate ~ factor(college$Private), xlab = "Private", ylab = "Outstate")

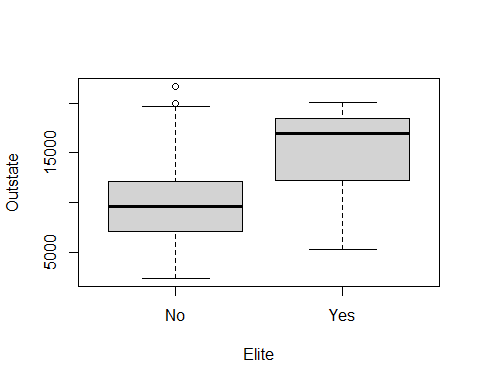


iv. Create a new qualitative variable, called `Elite`, by \_binning\_ the  
 `Top10perc` variable. We are going to divide universities into two  
 groups based on whether or not the proportion of students coming from  
 the top 10% of their high school classes exceeds 50%.  
   
 ```r  
 > Elite <- rep("No", nrow(college))  
 > Elite[college$Top10perc > 50] <- "Yes"  
 > Elite <- as.factor(Elite)  
 > college <- data.frame(college, Elite)  
 ```  
   
 Use the `summary()` function to see how many elite universities there  
 are. Now use the `plot()` function to produce side-by-side boxplots of  
 `Outstate` versus `Elite`.

college$Elite <- factor(ifelse(college$Top10perc > 50, "Yes", "No"))  
summary(college$Elite)

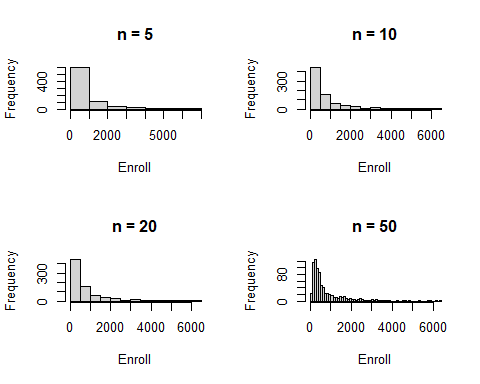
## No Yes   
## 699 78

plot(college$Outstate ~ college$Elite, xlab = "Elite", ylab = "Outstate")



v. Use the `hist()` function to produce some histograms with differing  
 numbers of bins for a few of the quantitative variables. You may find  
 the command `par(mfrow=c(2,2))` useful: it will divide the print   
 window into four regions so that four plots can be made   
 simultaneously. Modifying the arguments to this function will divide  
 the screen in other ways.

par(mfrow = c(2, 2))  
for (n in c(5, 10, 20, 50)) {  
 hist(college$Enroll, breaks = n, main = paste("n =", n), xlab = "Enroll")  
}



vi. Continue exploring the data, and provide a brief summary of what you  
 discover.

chisq.test(college$Private, college$Elite)

##   
## Pearson's Chi-squared test with Yates' continuity correction  
##   
## data: college$Private and college$Elite  
## X-squared = 4.3498, df = 1, p-value = 0.03701

### 

### NOMOR 9

1. Which of the predictors are quantitative, and which are qualitative?

auto <- read.csv("Auto.csv")  
auto <- na.omit(auto)

Kuantitatif : mpg, silinder, perpindahan, tenaga kuda, berat, tahun dan percepatan Kualitatif : asal dan nama

1. What is the range of each quantitative predictor? You can answer this using the range() function.

#apply(auto,2,range)  
sapply(auto[,1:7],range)

## mpg cylinders displacement horsepower weight acceleration year  
## [1,] "9" "3" "68" "?" "1613" "8" "70"  
## [2,] "46.6" "8" "455" "98" "5140" "24.8" "82"

1. What is the mean and standard deviation of each quantitative predictor?

sapply(auto[, 1:7], mean)

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA

## mpg cylinders displacement horsepower weight acceleration   
## 23.515869 5.458438 193.532746 NA 2970.261965 15.555668   
## year   
## 75.994962

sapply(auto[, 1:7], sd)

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =  
## na.rm): NAs introduced by coercion

## mpg cylinders displacement horsepower weight acceleration   
## 7.825804 1.701577 104.379583 NA 847.904119 2.749995   
## year   
## 3.690005

1. Now remove the 10th through 85th observations. What is the range, mean, and standard deviation of each predictor in the subset of the data that remains?

auto\_new=auto[-(10:85),]  
 #Auto[-(10:85),]  
  
apply(auto\_new,2,range)

## mpg cylinders displacement horsepower weight acceleration year origin  
## [1,] "11.0" "3" " 68" "?" "1649" " 8.5" "70" "1"   
## [2,] "46.6" "8" "455" "98" "4997" "24.8" "82" "3"   
## name   
## [1,] "amc ambassador brougham"  
## [2,] "vw rabbit custom"

sapply(auto\_new[, 1:7], mean)

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA

## mpg cylinders displacement horsepower weight acceleration   
## 24.438629 5.370717 187.049844 NA 2933.962617 15.723053   
## year   
## 77.152648

sapply(auto\_new[, 1:7], sd)

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm =  
## na.rm): NAs introduced by coercion

## mpg cylinders displacement horsepower weight acceleration   
## 7.908184 1.653486 99.635385 NA 810.642938 2.680514   
## year   
## 3.111230

1. Using the full data set, investigate the predictors graphically, using scatterplots or other tools of your choice. Create some plots highlighting the relationships among the predictors. Comment on your findings.

library(ggplot2)

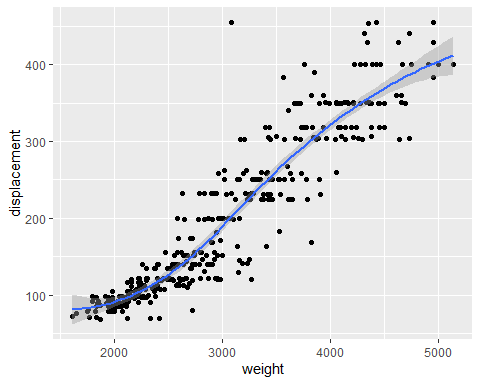
## Warning: package 'ggplot2' was built under R version 4.3.3

##   
## Attaching package: 'ggplot2'

## The following object is masked from 'Auto':  
##   
## mpg

ggplot(auto,aes(weight,displacement))+geom\_point()+geom\_smooth()

## `geom\_smooth()` using method = 'loess' and formula = 'y ~ x'

 Perpindahan dan berat berkorelasi positif dengan beberapa hubungan bersifat non-linier.

1. Suppose that we wish to predict gas mileage (mpg) on the basis of the other variables. Do your plots suggest that any of the other variables might be useful in predicting mpg? Justify your answer.

Dari plot di soal e, hampir semua prediktor berkorelasi dengan mpg kecuali nama.

### 

### NOMOR 10

1. To begin, load in the Boston data set. The Boston data set is part of the ISLR2 library in R.

library(ISLR2)

## Warning: package 'ISLR2' was built under R version 4.3.3

##   
## Attaching package: 'ISLR2'

## The following object is masked \_by\_ '.GlobalEnv':  
##   
## Auto

head(Boston)

## crim zn indus chas nox rm age dis rad tax ptratio lstat medv  
## 1 0.00632 18 2.31 0 0.538 6.575 65.2 4.0900 1 296 15.3 4.98 24.0  
## 2 0.02731 0 7.07 0 0.469 6.421 78.9 4.9671 2 242 17.8 9.14 21.6  
## 3 0.02729 0 7.07 0 0.469 7.185 61.1 4.9671 2 242 17.8 4.03 34.7  
## 4 0.03237 0 2.18 0 0.458 6.998 45.8 6.0622 3 222 18.7 2.94 33.4  
## 5 0.06905 0 2.18 0 0.458 7.147 54.2 6.0622 3 222 18.7 5.33 36.2  
## 6 0.02985 0 2.18 0 0.458 6.430 58.7 6.0622 3 222 18.7 5.21 28.7

dim(Boston)

## [1] 506 13

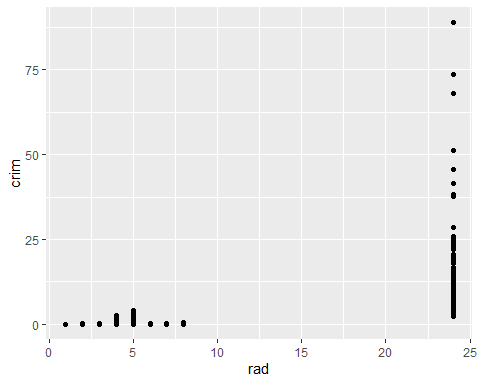
?Boston

How many rows are in this data set? How many columns? What do the rows and columns represent?

Ada 506 baris dan 14 kolom. Setiap baris mewakili satu rumah beserta atributnya. Setiap kolom mewakili satu set atribut rumah.

1. Make some pairwise scatterplots of the predictors (columns) in this data set. Describe your findings.

library(ggplot2)  
ggplot(Boston,aes(rad,crim))+geom\_point()

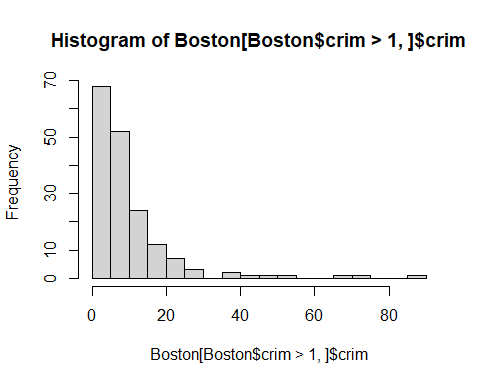
 Semakin tinggi indeks akses ke jalan raya radial, semakin tinggi pula kejahatan.

1. Are any of the predictors associated with per capita crime rate? If so, explain the relationship.

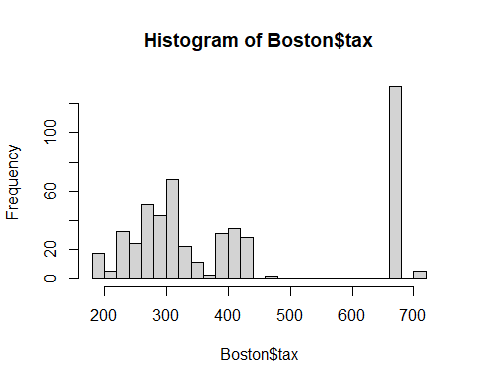
Ya, semakin tinggi indeks akses ke jalan raya radial, semakin tinggi pula kejahatan.

1. Do any of the census tracts of Boston appear to have particularly high crime rates? Tax rates? Pupil-teacher ratios? Comment on the range of each predictor.

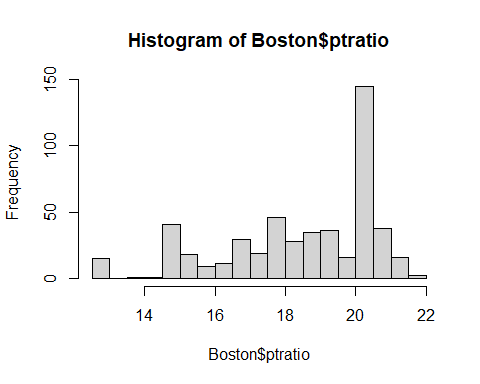
hist(Boston[Boston$crim>1,]$crim, breaks=25)



hist(Boston$tax, breaks=25)



hist(Boston$ptratio, breaks=25)

 Tingkat kejahatan : sebagian besar kota memiliki tingkat kejahatan yang rendah, tetapi beberapa daerah pinggiran kota memiliki tingkat kejahatan lebih dari 20 bahkan ada yang diatas 80.

Pajak : ada kesenjangan antara kota berpajak rendah dengan yang tinggi, puncaknya sekitar 0,5%.

Rasio murid-guru : tidak seimbang

1. How many of the census tracts in this data set bound the Charles river?

sum(Boston$chas)

## [1] 35

Ada 35 daerah pinggiran kota yang berbatasan dengan sungai Charles.

1. What is the median pupil-teacher ratio among the towns in this data set?

median(Boston$ptratio)

## [1] 19.05

Mediannya 19.05

1. Which census tract of Boston has lowest median value of owner-occupied homes? What are the values of the other predictors for that census tract, and how do those values compare to the overall ranges for those predictors? Comment on your findings.

subset(Boston,medv==min(Boston$medv))

## crim zn indus chas nox rm age dis rad tax ptratio lstat medv  
## 399 38.3518 0 18.1 0 0.693 5.453 100 1.4896 24 666 20.2 30.59 5  
## 406 67.9208 0 18.1 0 0.693 5.683 100 1.4254 24 666 20.2 22.98 5

Untuk rumah yang dimiliki pemiliknya, sub-urban ke-399 dan ke-406 memiliki nilai rata-rata terendah.

1. In this data set, how many of the census tract average more than seven rooms per dwelling? More than eight rooms per dwelling? Comment on the census tracts that average more than eight rooms per dwelling.

sum(Boston$rm > 7)

## [1] 64

sum(Boston$rm > 8)

## [1] 13

summary(Boston)

## crim zn indus chas   
## Min. : 0.00632 Min. : 0.00 Min. : 0.46 Min. :0.00000   
## 1st Qu.: 0.08205 1st Qu.: 0.00 1st Qu.: 5.19 1st Qu.:0.00000   
## Median : 0.25651 Median : 0.00 Median : 9.69 Median :0.00000   
## Mean : 3.61352 Mean : 11.36 Mean :11.14 Mean :0.06917   
## 3rd Qu.: 3.67708 3rd Qu.: 12.50 3rd Qu.:18.10 3rd Qu.:0.00000   
## Max. :88.97620 Max. :100.00 Max. :27.74 Max. :1.00000   
## nox rm age dis   
## Min. :0.3850 Min. :3.561 Min. : 2.90 Min. : 1.130   
## 1st Qu.:0.4490 1st Qu.:5.886 1st Qu.: 45.02 1st Qu.: 2.100   
## Median :0.5380 Median :6.208 Median : 77.50 Median : 3.207   
## Mean :0.5547 Mean :6.285 Mean : 68.57 Mean : 3.795   
## 3rd Qu.:0.6240 3rd Qu.:6.623 3rd Qu.: 94.08 3rd Qu.: 5.188   
## Max. :0.8710 Max. :8.780 Max. :100.00 Max. :12.127   
## rad tax ptratio lstat   
## Min. : 1.000 Min. :187.0 Min. :12.60 Min. : 1.73   
## 1st Qu.: 4.000 1st Qu.:279.0 1st Qu.:17.40 1st Qu.: 6.95   
## Median : 5.000 Median :330.0 Median :19.05 Median :11.36   
## Mean : 9.549 Mean :408.2 Mean :18.46 Mean :12.65   
## 3rd Qu.:24.000 3rd Qu.:666.0 3rd Qu.:20.20 3rd Qu.:16.95   
## Max. :24.000 Max. :711.0 Max. :22.00 Max. :37.97   
## medv   
## Min. : 5.00   
## 1st Qu.:17.02   
## Median :21.20   
## Mean :22.53   
## 3rd Qu.:25.00   
## Max. :50.00

64 pinggiran kota memiliki rata-rata lebih dari 7 kamar, dan 13 pinggiran kota memiliki rata-rata lebih dari 8 kamar.