2.R_Introduction to R

R notebook using data from 2.R_Introduction to R_auto.csv · 4 views · Edit tags

In [1]:

```
## Importing packages

# This R environment comes with all of CRAN and many other
 helpful packages preinstalled.
# You can see which packages are installed by checking out
 the kaggle/rstats docker image:
# https://github.com/kaggle/docker-rstats

library(tidyverse) # metapackage with lots of helpful func
tions

## Running code

# In a notebook, you can run a single code cell by clickin
g in the cell and then hitting
# the blue arrow to the left, or by clicking in the cell a
nd pressing Shift+Enter. In a script,
# you can run code by highlighting the code you want to ru
n and then clicking the blue arrow
# at the bottom of this window.

## Reading in files

# You can access files from datasets you've added to this
 kernel in the "../input/" directory.
# You can see the files added to this kernel by running th
e code below.

list.files(path = "../input")

## Saving data

# If you save any files or images, these will be put in th
e "output" directory. You
# can see the output directory by committing and running y
our kernel (using the
# Commit & Run button) and then checking out the compiled
 version of your kernel.
```

```
── Attaching packages ─────────────────────
──────────────── tidyverse 1.2.1 ──
✔ ggplot2 3.0.0.9000     ✔ purrr   0.2.5
```

```
✔ tibble  1.4.2         ✔ dplyr   0.7.6
✔ tidyr   0.8.1         ✔ stringr 1.3.1
✔ readr   1.2.0         ✔ forcats 0.3.0
── Conflicts ──────────────────────────────────
──────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
```

'Auto.csv'

In [2]:

```
## About this notebook
# StatLearning - SELF PACED Statistical Learning
# https://github.com/JunChiehWang/Statistical_Learning_Stanford
# https://youtu.be/jwBgGS_4RQA
```

In [3]:

```
# So here we assign three numbers to a vector x.
# create a vector
x=c(2,7,5)

# And if we just type in x, it will print the vector.
x
```

2 7 5

In [4]:

```
# there are many way to create vectors, here's another way,
# making a sequence, starting from 4, and having length 3,
 and in steps by 3.
y=seq(from=4,length=3,by=3)
y
```

4 7 10

In [5]:

```
# And if you want to find out more about it, in R the way
 to
# get help on functions and objects is to put a question
# mark in front.
#
# these 3 works !
# ?seq
# help(seq)
?seq()
```

In [6]:

```
y
```

4 7 10

In [7]:

```
# R does vector operations in parallel.
# So if we say x plus y, even though they're both vectors,
# we get the sum of those two vectors, element by element.
x+y
```

6 14 15

In [8]:

```
# And likewise, other operations, like x/y means x,
# divide y, and it does element-wise
# division of the elements.
x/y
```

0.5 1 0.5

In [9]:

```
# And you can do x to the power y, and it'll do element-wise exponentiation.
x^y
```

```
16  823543  9765625
```

In [10]:

```
# What about accessing elements of a vectors?
# Well, so there's a subscript convention in R using square braces.
x[2]
```

```
7
```

In [11]:

```
# If we go x square brace 2 colon 3, that says we want the
# elements of x starting from element 2 and
# ending at element 3.
x[2:3]
```

```
7  5
```

In [12]:

```
# So x minus 2 means remove the element 2 from x, and return
# the subsetted vector.
x[-2]
```

```
2  5
```

In [13]:

```
# And you can remove more than one element at a time.
# And so here we're moving the collection of indices 1 and 2,
# and they can be arbitrary collection of indices.
# And that just gives us a vector of length 1.
```

```
# And that just gives us a vector of length 1.
x
x[-c(1,2)]

# Something to note, there's no scalars in R.
# Everything's a vector.
# So a scalar is just a vector of length 1.
```

```
2  7  5
```

```
5
```

In [14]:

```
# in R. A matrix is a two-way array.
# And here's a simple way of making a matrix.
# And we give it the numbers 1 to 12.
# So the first argument are the actual numbers in the matr
ix.
# And then we give it the dimensions 4 and 3.
# So we want to make a 4 by 3 matrix.
z=matrix(seq(1,12),4,3)
z
#And so you see it's taken the numbers in column order,
```

1	5	9
2	6	10
3	7	11
4	8	12

In [15]:

```
# we can subset elements of a matrix.
# So here we want to see the third and fourth row, and the
# second and third column.
z[3:4,2:3]
```

7	11
8	12

In [16]:

```
# And if just put a comma and ignore the first index, yo
u'll
# just get the columns.
z[,2:3]
```

5	9
6	10
7	11
8	12

In [17]:

```
#And there is the first column of z.
z[,1]
# Now notice what's happened.
# When we took just the first column of z, that became a
# vector, and it actually dropped its matrix status.
```

1 2 3 4

In [18]:

```
# Sometimes that's convenient.
# But a lot of the time, it's not, especially when your
# programming and you don't want to accidentally lose the
# status of a matrix.
# So the matrix subsetting has an argument, drop, and here
 we
# say, drop equals false, and it keeps that one column mat
rix
# as a matrix, and not a vector.
z[,1,drop=FALSE]
```

1
2
3

```
4
```

In [19]:

```
# You could query the dimension of a matrix.
# So dim of z gives you the dimensions of the matrix.
dim(z)
dim(z[,1])
dim(z[,1,drop=FALSE])
```

4 3

NULL

4 1

In [20]:

```
# ls is a very nice command.
# It tells you what you have available in
# your working directory.
ls()
```

'x' 'y' 'z'

In [21]:

```
# You can clean up your working directory.
# So for example, you can use the rm command to remove y.
# And there we see y is gone.
rm(y)
ls()
```

'x' 'z'

In [22]:

```
# this command will create 50 random uniforms on 01.
x=runif(50)
x
```

0.305004990659654 0.90372620546259
0.125520611414686 0.40733418893069 0.26363483723253
0.700393505161628 0.896757421316579
0.360145977698267 0.0244052303023636
0.34604942961596 0.516782845370471
0.601661927765235 0.88921434036456 0.55240605189465
0.232825397048146 0.0822875432204455
0.51563825388439 0.637167218374088
0.804697640938684 0.537349166348577
0.951800716575235 0.0900391265749931
0.27483018185012 0.0858914230484515
0.756379132159054 0.156591568840668
0.689169563353062 0.854616188677028
0.805841657100245 0.936128148809075
0.727414100198075 0.628371731610969
0.304068616358563 0.89705546409823
0.870223144767806 0.158416997408494
0.277487333631143 0.395480957115069
0.0602524937130511 0.67742024618201
0.950397868640721 0.640315833501518
0.897917555179447 0.355640702182427
0.518099416512996 0.489226570120081
0.881468788487837 0.691429120954126
0.280485245399177 0.0392694636248052

In [23]:

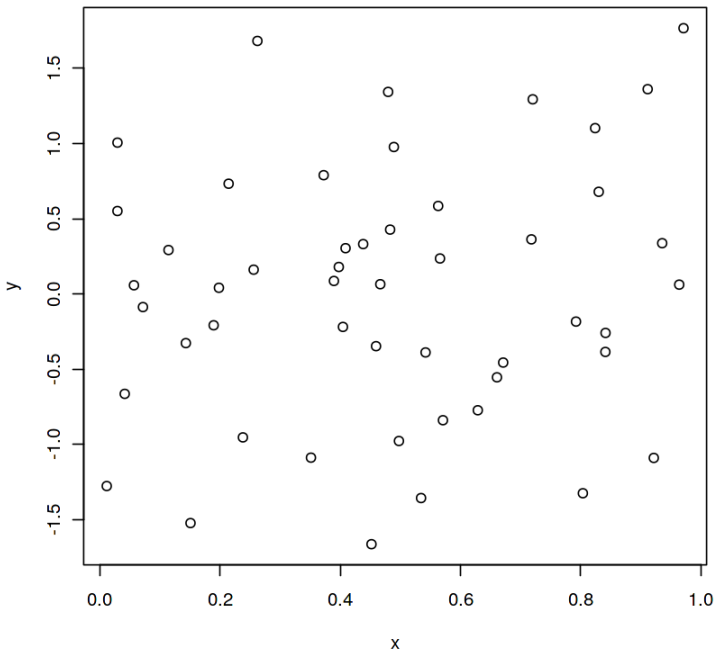
```
# And rnorm, is random norm, random Gaussians, random
# normal variables.
# It will create 50 standard random normal variables.
y=rnorm(50)
y
```

2.39909884020822 -1.12776614890965 1.37255704882152
-0.770611118147271 0.335701794075011
-0.69189928446017 0.299382224216859
-0.567541249028059 -0.315585573654434
0.416631115112537 0.113034770475831 1.36115170823749
0.0766575107175832 -0.389892379671173
-0.0629983355097977 -0.145951659918114

```
-1.11262095826317  0.30855169840351
-0.0539227760959545  0.423946739936007
1.2383949482972  -1.3334653636717  -0.0455403797514575
-0.937534325282983  1.36343942654727
0.348278504325598  -0.184122784312461
-1.05882055873751  0.586292435488043
-2.30043760670914  0.55397101531614
0.0355074143264037  0.514520582392639
1.39932410544702  0.422604864722667
0.332188294042047  -1.86619052390198
-0.410073023314603  -0.892054482386566
1.06526301804682  1.35938482040876  0.218519902923331
0.635380116359706  -1.18546503972369  1.4595212392717
2.16681278119597  0.739201222253092
-0.418122919293427  -1.1887723033706
-1.18519996225316
```

In [24]:

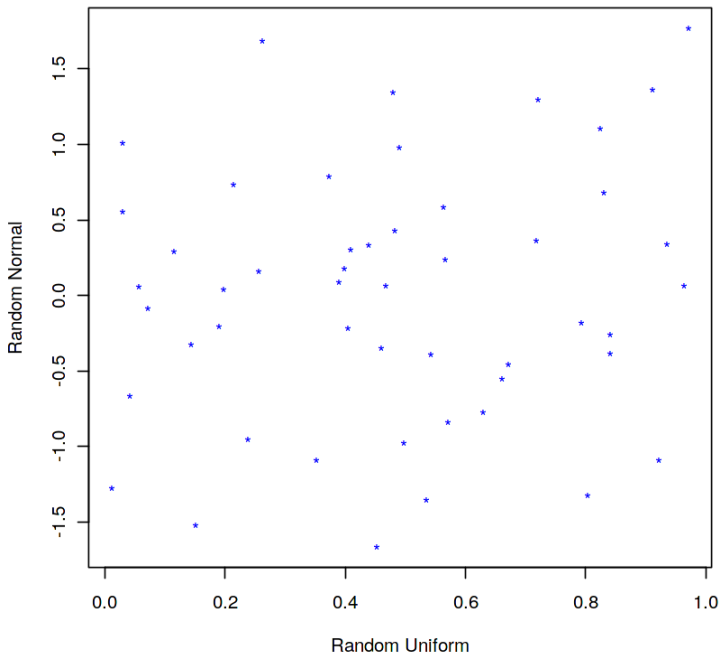
```
# plot x and y
x=runif(50)
y=rnorm(50)
plot(x,y)
```



In [25]:

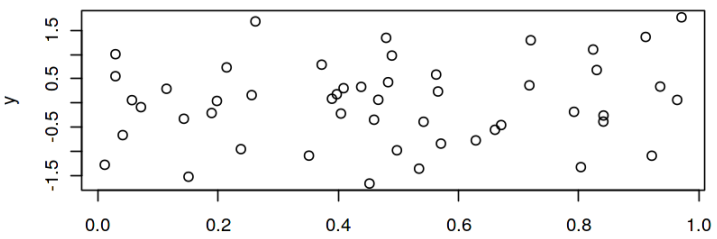
In [25]:

```
plot(x,y,xlab="Random Uniform",ylab="Random Normal",pch=
"*",col="blue")
```

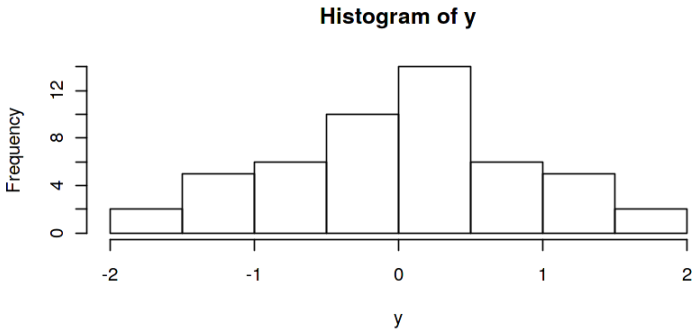


In [26]:

```
# And the par command allows you to set some of these opti
ons.
# Some you can do directly in the plot command, and some,
# like layout commands, you can set with par.
# So this is one that's often used, mf row.
# It says we want to have a panel of plots with two rows
# and one column.
# And so that we do with the mf row command.
par(mfrow=c(2,1))
plot(x,y)
hist(y)
```



x



In [27]:

```
# And so that mf row, that division will stay in place
# until you reset it with another mf row command.
par(mfrow=c(1,1))
```

In [28]:

```
# Excel is often the place where you store your data.
# And so we're going to read that.
# There's ways of doing this in R. So we use the read.csv
 function.
# And so this requires that you've saved your data in
# comma separated value from Excel.
# And then you can just read it in, in R, and it respects
 the
# rows, and columns, and the headings, and everything els
e.
getwd()
list.files(path = "../input")
Auto=read.csv("../input/Auto.csv")
```

'/kaggle/working'

'Auto.csv'

In [29]:

```
# And you can see it's got a number of columns.
# And those are the names of the variables.
names(Auto)
```

'mpg' 'cylinders' 'displacement' 'horsepower' 'weight' 'acceleration' 'year' 'origin' 'name'

In [30]:

```
# And we can look at the dimension of the data.
# It's 397 by 9.
dim(Auto)
```

397 9

In [31]:

```
# And we can see, what is this object that we read in?
# The class of Auto is dataframe.
class(Auto)

# And you'll learn more about dataframes.
# They're very valuable objects.
# It's sort of like a matrix, except that the columns can
 be
# variables of different kinds.
```

'data.frame'

In [32]:

```
# Summaries are a useful function for a dataframe.
# It'll give you a summary of each of the variables in the
 data frame.
# And you can see it's things like min, max, and so on.
# Horsepower, for example, is a categorical variable, so i
t
# actually gives you all the values.
# And the name of the automobile is also categorical.
# It's an effective variable.
# It gives you the values.
summary(Auto)
```

```
      mpg          cylinders      displaceme
```

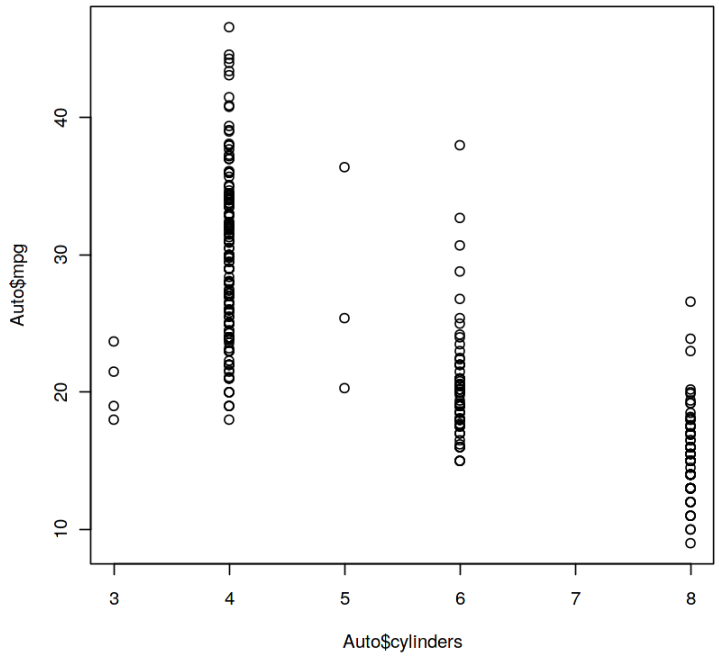
```
nt     horsepower      weight
 Min.   : 9.00   Min.   :3.000   Min.   : 6
8.0   150    : 22   Min.   :1613
 1st Qu.:17.50   1st Qu.:4.000   1st Qu.:10
4.0   90     : 20   1st Qu.:2223
 Median :23.00   Median :4.000   Median :14
6.0   88     : 19   Median :2800
 Mean   :23.52   Mean   :5.458   Mean   :19
3.5   110    : 18   Mean   :2970
 3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:26
2.0   100    : 17   3rd Qu.:3609
 Max.   :46.60   Max.   :8.000   Max.   :45
5.0   75     : 14   Max.   :5140

    (Other):287
  acceleration        year           origin
                name
 Min.   : 8.00   Min.   :70.00   Min.   :1.0
00   ford pinto    :  6
 1st Qu.:13.80   1st Qu.:73.00   1st Qu.:1.0
00   amc matador   :  5
 Median :15.50   Median :76.00   Median :1.0
00   ford maverick :  5
 Mean   :15.56   Mean   :75.99   Mean   :1.5
74   toyota corolla:  5
 3rd Qu.:17.10   3rd Qu.:79.00   3rd Qu.:2.0
00   amc gremlin   :  4
 Max.   :24.80   Max.   :82.00   Max.   :3.0
00   amc hornet    :  4

    (Other)       :368
```

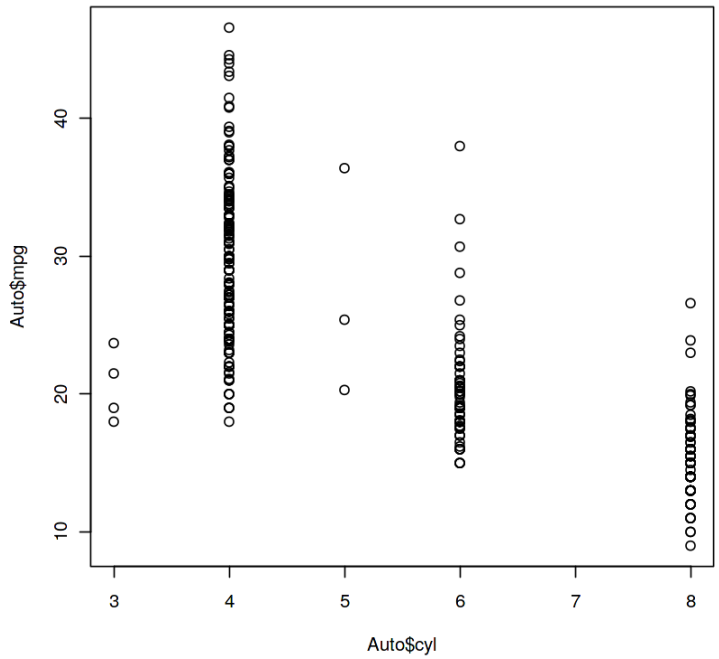
In [33]:

```
# Now you can plot the elements of a dataframe.
# So a dataframe is also a list.
# And a list, you get the elements of the list by giving
# the name of the list, which is Auto here, and then use d
ollar sign.
# And then you can give the name.
# So that's one way of getting the elements of a list.
plot(Auto$cylinders,Auto$mpg)
```



In [34]:

```
plot(Auto$cyl,Auto$mpg)
```



In [35]:

```
# So that's a little cumbersome, having to do that dollar
# indexing of the elements of the dataframe.
# So what you can actually do is you can attach the datafr
```

```
# So what you can actually do is you can attach the datafr
ame.
# And what it does, is it creates a workspace with all
# the named variables as now variables in your workspace.
# So now you can access them by name.
attach(Auto)
```

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

    mpg
```

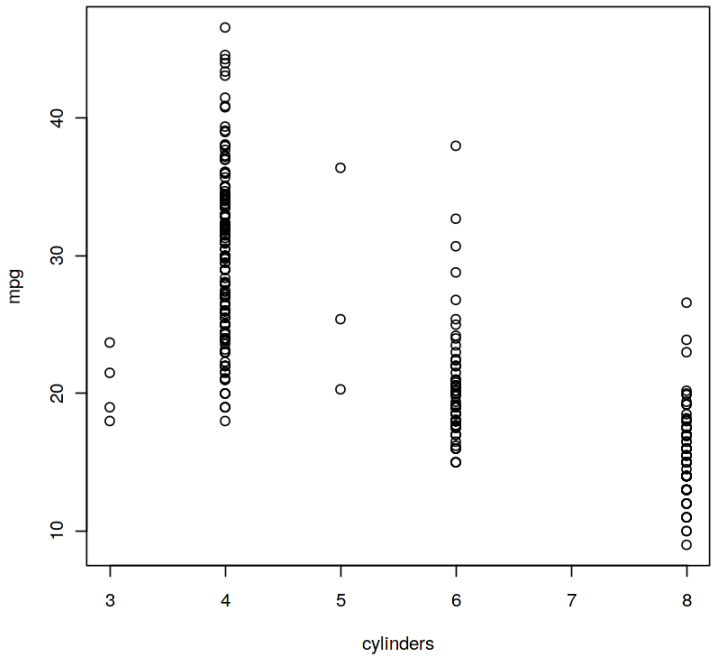
In [36]:

```
# And so if we do issue the command Search, it tells us
# our various workspaces.
# And there we see the global environment (.GlobalEnv) is
 where we've put
# all our vectors, like x, y, and z, and the variables w
e've
# created in the session.
# But this dataframe that we've attached is in the second
# position here. (Auto)
# And you'll see there's other things in the
# Search path, as well.
# And these are largely packages, at this point, whose fun
ctions we have available.
search()
```

'.GlobalEnv' 'Auto' 'package:forcats' 'package:stringr' 'package:dplyr' 'package:purrr' 'package:readr' 'package:tidyr' 'package:tibble' 'package:ggplot2' 'package:tidyverse' 'jupyter:irkernel' 'package:stats' 'package:graphics' 'package:grDevices' 'package:utils' 'package:datasets' 'package:methods' 'Autoloads' 'package:base'

In [37]:

```
# So now we can do that plot command more directly.
plot(cylinders,mpg)
```



In [38]:

```
# The function factor is used to encode a vector as a fact
or
# (the terms 'category' and 'enumerated type' are also use
d for factors).
?as.factor

cylinders

as.factor(cylinders)

cylinders=as.factor(cylinders)
cylinders
```

```
8 8 8 8 8 8 8 8 8 8 8 8 8 8 4 6 6 6 4
4 4 4 4 4 6 8 8 8 8 4 4 4 4 6 6 6 6 6
8 8 8 8 8 8 8 6 4 6 6 4 4 4 4 4 4 4 4
4 4 4 4 4 8 8 8 8 8 8 8 8 8 3 8 8 8 8
4 4 4 4 4 4 4 4 4 8 8 8 8 8 8 8 8 8 8
8 8 6 6 6 6 6 4 8 8 8 8 6 4 4 4 3 4 6
4 8 8 4 4 4 4 8 4 6 8 6 6 6 6 4 4 4 4
6 6 6 8 8 8 8 8 4 4 4 4 4 4 4 4 4 4 4
6 6 6 6 8 8 8 8 6 6 6 6 6 8 8 4 4 6 4
4 4 4 6 4 6 4 4 4 4 4 4 4 4 4 4 8 8 8
8 6 6 6 6 4 4 4 4 6 6 6 6 4 4 4 4 4 8
```

```
4 6 6 8 8 8 8 4 4 4 4 4 8 8 8 8 6 6 6
6 8 8 8 8 4 4 4 4 4 4 4 4 6 4 3 4 4 4
4 4 8 8 8 6 6 6 4 6 6 6 6 6 6 8 6 8 8
4 4 4 4 4 4 4 4 5 6 4 6 4 4 6 6 4 6 6
8 8 8 8 8 8 8 8 4 4 4 4 5 8 4 8 4 4 4
4 4 6 6 4 4 4 4 4 4 4 4 6 4 4 4 4 4 4
4 4 4 4 5 4 4 4 4 4 6 3 4 4 4 4 4 4 6
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 6
6 6 6 8 6 6 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 6 6 4 6 4 4 4 4 4 4 4 4
```

```
8 8 8 8 8 8 8 8 8 8 8 8 8 8 4 6 6 6 4
4 4 4 4 4 6 8 8 8 8 4 4 4 4 6 6 6 6 6
8 8 8 8 8 8 8 6 4 6 6 4 4 4 4 4 4 4 4
4 4 4 4 4 8 8 8 8 8 8 8 8 8 3 8 8 8 8
4 4 4 4 4 4 4 4 4 8 8 8 8 8 8 8 8 8 8
8 8 6 6 6 6 6 4 8 8 8 8 6 4 4 4 3 4 6
4 8 8 4 4 4 4 8 4 6 8 6 6 6 6 4 4 4 4
6 6 6 8 8 8 8 8 4 4 4 4 4 4 4 4 4 4 4
6 6 6 6 8 8 8 8 6 6 6 6 6 8 8 4 4 6 4
4 4 4 6 4 6 4 4 4 4 4 4 4 4 4 4 8 8 8
8 6 6 6 6 4 4 4 4 6 6 6 6 4 4 4 4 4 8
4 6 6 8 8 8 8 4 4 4 4 4 8 8 8 8 6 6 6
6 8 8 8 8 4 4 4 4 4 4 4 4 6 4 3 4 4 4
4 4 8 8 8 6 6 6 4 6 6 6 6 6 6 8 6 8 8
4 4 4 4 4 4 4 4 5 6 4 6 4 4 6 6 4 6 6
8 8 8 8 8 8 8 8 4 4 4 4 5 8 4 8 4 4 4
4 4 6 6 4 4 4 4 4 4 4 4 6 4 4 4 4 4 4
4 4 4 4 5 4 4 4 4 4 6 3 4 4 4 4 4 4 6
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 6
6 6 6 8 6 6 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 6 6 4 6 4 4 4 4 4 4 4 4
```

▶ **Levels**:

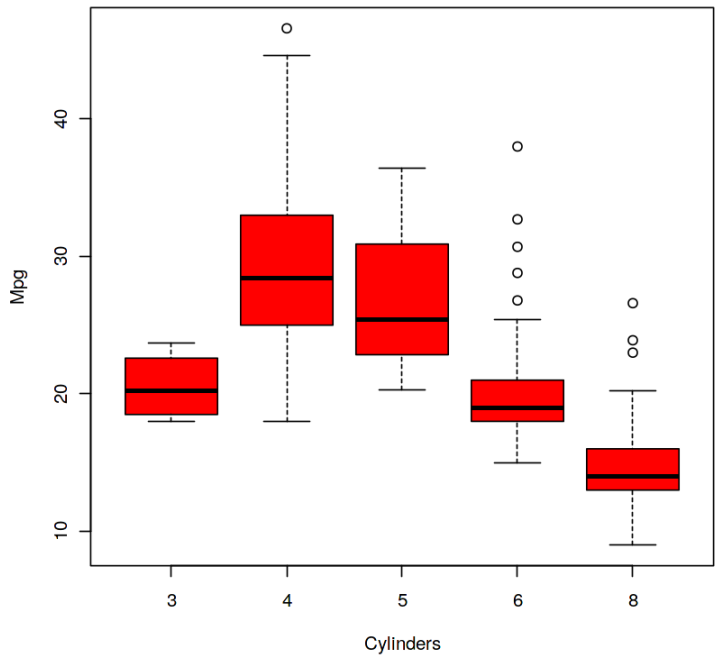
```
8 8 8 8 8 8 8 8 8 8 8 8 8 8 4 6 6 6 4
4 4 4 4 4 6 8 8 8 8 4 4 4 4 6 6 6 6 6
8 8 8 8 8 8 8 6 4 6 6 4 4 4 4 4 4 4 4
4 4 4 4 4 8 8 8 8 8 8 8 8 8 3 8 8 8 8
4 4 4 4 4 4 4 4 4 8 8 8 8 8 8 8 8 8 8
8 8 6 6 6 6 6 4 8 8 8 8 6 4 4 4 3 4 6
4 8 8 4 4 4 4 8 4 6 8 6 6 6 6 4 4 4 4
6 6 6 8 8 8 8 8 4 4 4 4 4 4 4 4 4 4 4
6 6 6 6 8 8 8 8 6 6 6 6 6 8 8 4 4 6 4
4 4 4 6 4 6 4 4 4 4 4 4 4 4 4 4 8 8 8
```

```
8 6 6 6 6 4 4 4 4 6 6 6 6 4 4 4 4 4 8
4 6 6 8 8 8 8 4 4 4 4 4 8 8 8 8 6 6 6
6 8 8 8 8 4 4 4 4 4 4 4 4 6 4 3 4 4 4
4 4 8 8 8 6 6 6 4 6 6 6 6 6 6 8 6 8 8
4 4 4 4 4 4 4 4 5 6 4 6 4 4 6 6 4 6 6
8 8 8 8 8 8 8 8 4 4 4 4 5 8 4 8 4 4 4
4 4 6 6 4 4 4 4 4 4 4 4 6 4 4 4 4 4 4
4 4 4 4 5 4 4 4 4 4 6 3 4 4 4 4 4 4 6
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 6
6 6 6 8 6 6 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 6 6 4 6 4 4 4 4 4 4 4 4
```

▶ **Levels**:

In [39]:

```
plot(cylinders,mpg,xlab="Cylinders",ylab="Mpg",col="red")
```



In [40]:

```
# pdf starts the graphics device driver for producing PDF
 graphics.
# ?pdf
pdf(file="./mpg.pdf")
list.files(path = "./")
```

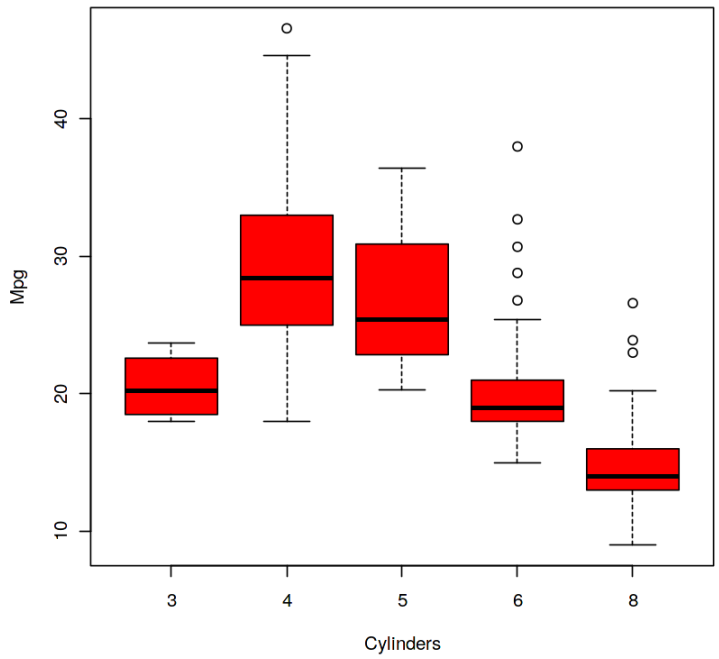
'__output__.json' 'mpg.pdf' 'Rplot001.png' 'script.irnb'

In [41]:

```
plot(cylinders,mpg,xlab="Cylinders",ylab="Mpg",col="red")

# dev.off shuts down the specified (by default the curren
t) device. I
# ?dev.off()
dev.off()
```

pdf: 3



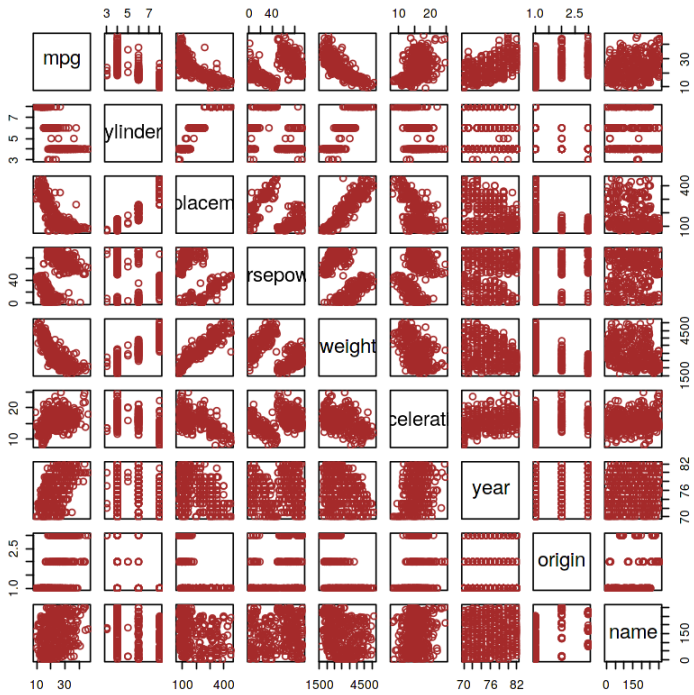
In [42]:

```
list.files(path = "./")
```

'__output__.json' 'mpg.pdf' 'Rplot001.png' 'script.irnb'

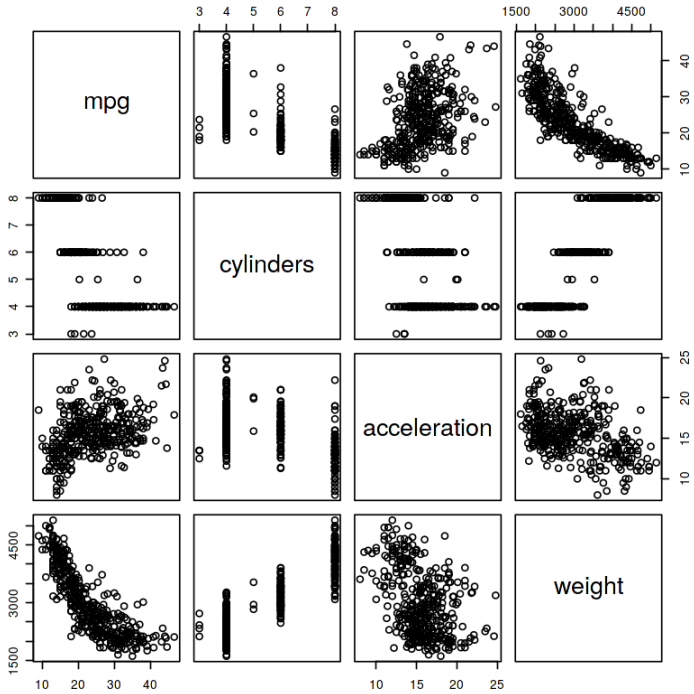
In [43]:

```
pairs(Auto,col="brown")
```



In [44]:

```
pairs(mpg~cylinders+acceleration+weight,Auto)
```



In [45]:

```
#The function quit or its alias q terminate the current R
 session.
#?q()
q()
```

In [ ]:

```

```


Did you find this Kernel useful?
Show your appreciation with an upvote

0

Data

Data Sources

2.R_Introducti...

... 397 x 9

2.R_Introduction to R_auto.csv

Last Updated: 4 days ago (Version 1)

About this Dataset

No description yet

Output Files

New Dataset | New Kernel | Download All

Output Files

mpg.pdf

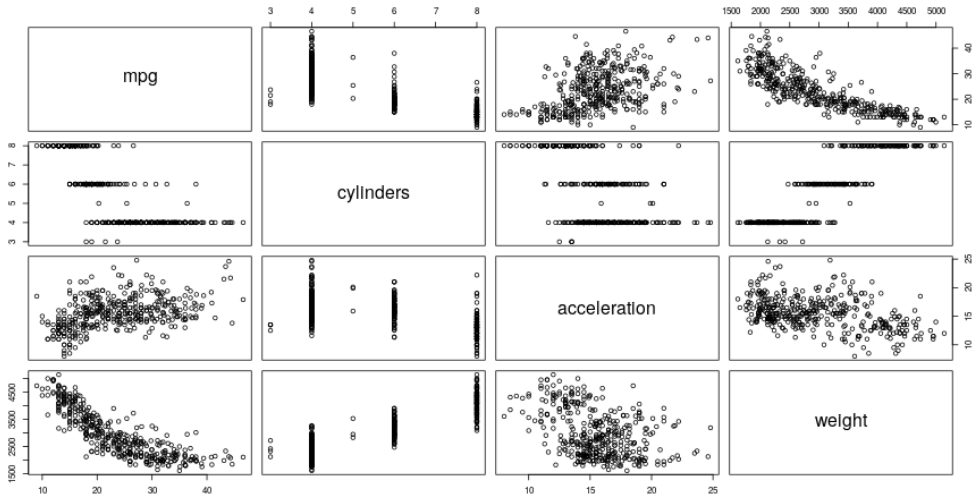
About this file

This file was created from a Kernel, it does not have a description.

mpg.pdf

We don't support previews for this file yet

Output Visualizations



Run Info

Succeeded	True	Run Time	11.6 secc

Exit Code	0	Queue Time	0 second
Docker Image Name	kaggle/rstats (Dockerfile)	Output Size	0
Timeout Exceeded	False	Used All Space	False
Failure Message			

Log

Download Log

```
Time  Line #  Log Message
2.1s       1  [NbConvertApp] Converting notebook script.irnb
              to html
4.3s       2  [NbConvertApp] Executing notebook with kernel:
              ir
11.3s      3  [NbConvertApp] Support files will be in
              __results___files/
              [NbConvertApp] Making directory
              __results___files
              [NbConvertApp] Making directory
              __results___files
11.3s      4  [NbConvertApp] Making directory
              __results___files
              [NbConvertApp] Making directory
              __results___files
              [NbConvertApp] Making directory
              __results___files
              [NbConvertApp] Making directory
              __results___files
              [NbConvertApp] Making directory
              __results___files
              [NbConvertApp] Making directory
              __results___files
              [NbConvertApp] Making directory
              __results___files
              [NbConvertApp] Making directory
              __results___files
              [NbConvertApp] Writing 337949 bytes to
              __results__.html
11.3s      5
11.3s      7  Complete. Exited with code 0.
```

Comments (0)

Click here to enter a comment...