Correlation using R langauge

Author: Sameh MAGDELDIN

Module contents:

- Correlation (Pearson, Spearman,and kendall)
- Correlation matrix

Packages needed in this tutorial

```
library("MASS")  # this datasets is found in MASS package or u can load it directly
library("ggplot2") # for plotting
library("GGally") # an extention for ggplot2
library("car") # for some graphical and analytical functions
library("PerformanceAnalytics")# we will need this package for correlation later
library("plyr") #data manipulation
library("dplyr") #data manipulation "should be installed after plyr"
library("RColorBrewer") #this is to give a nice coloring for ur figures
```

Correlation

Bivariate analysis that measures the strengths of association between two variable. Correlation, in its most common form, is a measure of linear dependence; the catch is that not all dependencies are linear. Correlation implies dependence but not the reverse.

Role of Thumb [Correlation does not mean causation]

####In this tutorial, we will learn how to perform correlation and regression analysis

####I will use the cats dataset,, I LOVE CATS!

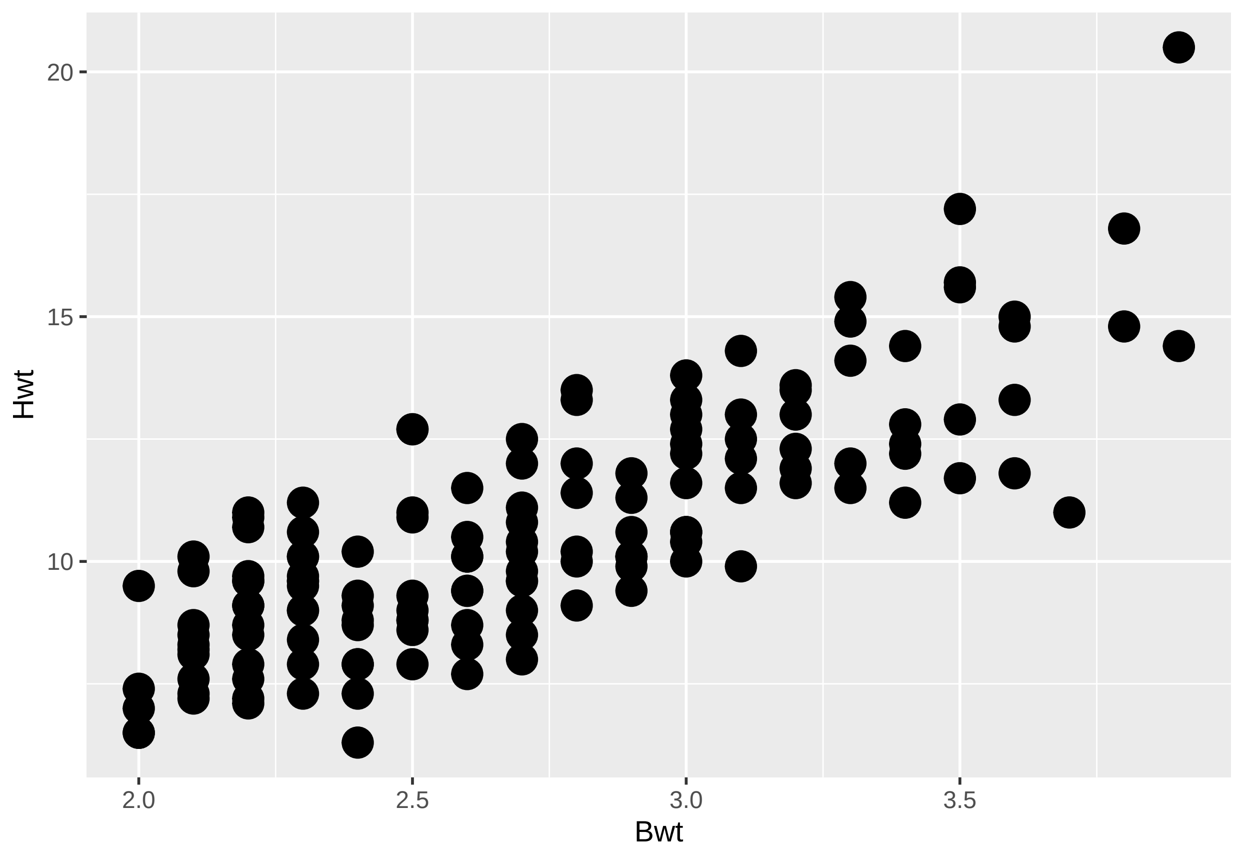
####lets have a look on the cats dataset

```
head(cats)
summary(cats)
```

This simple dataset reports the observation of cats body weight (kg) versus their heart weight (g)

###First, lets scatter plot the cat body weight versus their heart weight

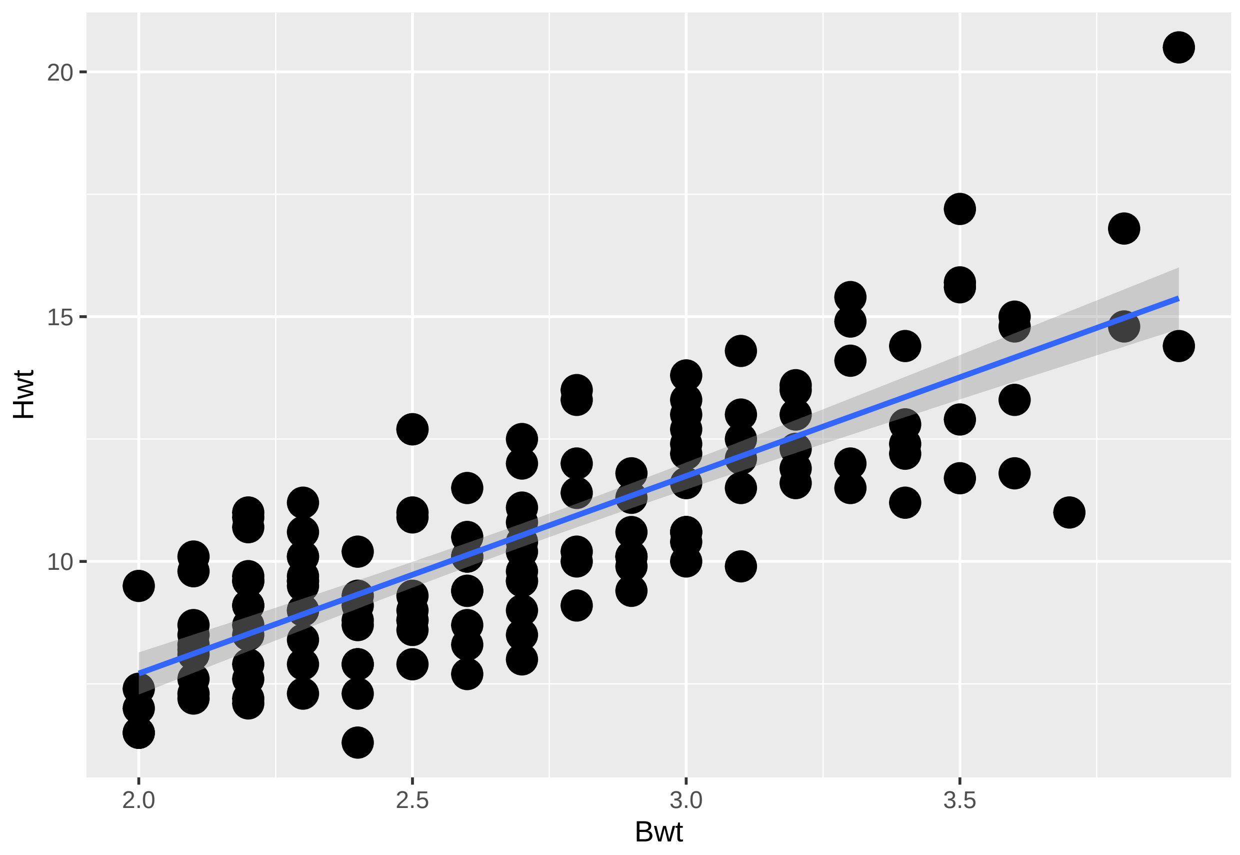
```
ggplot(cats, aes(x=Bwt, y=Hwt)) + geom_point(size=5)
```



as straight forward, check the correlation first using this ggplot code

```r
ggplot(cats, aes(x=Bwt, y=Hwt)) +geom_point(size=5)+geom_smooth(method=lm)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

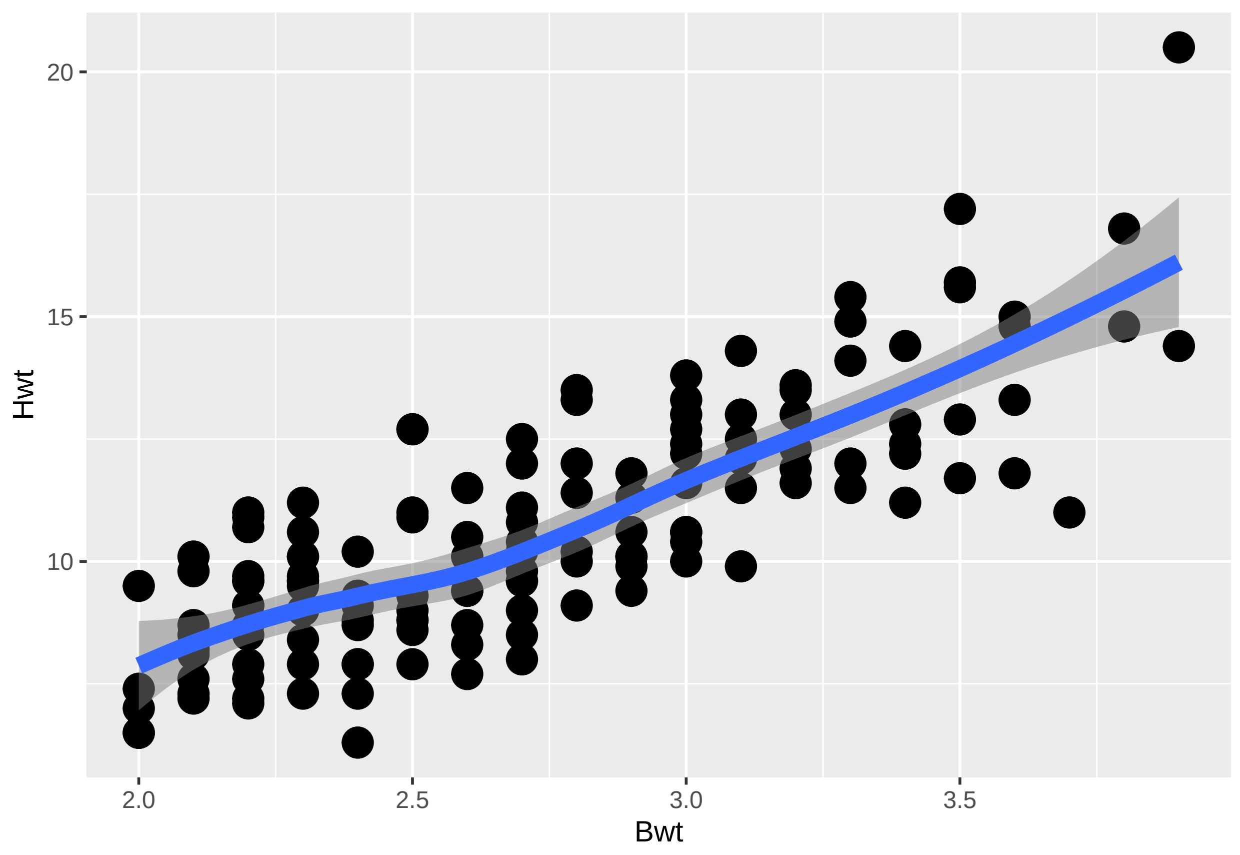


in ggplot, you can control every thing. try this

```
ggplot(cats, aes(x=Bwt, y=Hwt))+ geom_point(size=5)+geom_smooth(fill = "grey50", size = 3, alpha = 0.5)
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
## ‘geom_smooth()‘ using method = ’loess’ and formula = ’y ~ x’
```



Because u did not set the method of adding line, R set the default "loess" [(LOcally wEighted Scatter-plot Smoother)]

Smoothing method (function) to use, accepts either NULL or e.g. "lm", "glm", "gam", "loess" or a function, e.g. MASS::rlm or mgcv::gam, stats::lm, or stats::loess.

###The scatterplot shows reasonable linear relationship between the two variables. I can say that becasue the fitted line is raised up. However, it is not directed to the upper right corner.

###Lets run the correlation to confirm that pattern.

Use cor function or cor.test function

> Before that? Which test to use? Pearson for parametric data, spearman and kendall for non parametric. if your data have some outliers that you suspect use non parametric test.

```
cor(cats$Bwt,cats$Hwt,method="spearman")
```

```
## [1] 0.7908427
```

```
# spearman test for non parametric analysis.
# Check the normality test to see if your data falls under guassian distribution (normal; parametric) o
```

```r
cor(cats$Bwt,cats$Hwt,method="pearson")# for parametric test
```

```
## [1] 0.8041274
```

```r
cor(cats$Bwt,cats$Hwt,method="kendall") # non parameteric
```

```
## [1] 0.6079403
```

> Spearman and Kendall „ yes i know what you want to say. which one to use? Usually using Spearman is more popular than kendall. confidence intervals for Spearman are less reliable and less interpretable than confidence intervals for Kendalls, according to Kendall & Gibbons (1990).

```r
cor.test(cats$Bwt,cats$Hwt)
```

```
##
##  Pearson's product-moment correlation
##
## data:  cats$Bwt and cats$Hwt
## t = 16.119, df = 142, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.7375682 0.8552122
## sample estimates:
##       cor
## 0.8041274
```

```r
# as u see default is pearson
```

```r
cor.test(cats$Bwt,cats$Hwt,method="spearman",exact = FALSE)
```

```
##
##  Spearman's rank correlation rho
##
## data:  cats$Bwt and cats$Hwt
## S = 104085, p-value < 2.2e-16
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.7908427
```

```r
# "exact"" if you want to examine a specific p value.
# Here and in general we don't use it so default is FALSE
```

what is the difference?, cor returns correlation coefficient (r) only , while cor.test gives both (r) and p value.

*** Data interpretation ***

as you can see, the *correlation coefficient (r)* is close to 1 indicating a strong positive correlationship between cats body weight and their heart weight.

> ***Hint1***: r close to 1 means positive correlation and close to -1.0 indicates negative correlation, while around 0 means no significant correlation can be seen.
>
> ***Hint2*** correlation coefficient (r)is some times referred as "Pearson product moment correlation coefficient"
>
> ***Hint3*** A correlation greater than 0.8 is generally described as strong, whereas a correlation less than 0.5 is generally described as weak.
>
> ***Hint4*** Don't confuse between r and r2 (R2); r2 (R2) is called ***(Coefficient of Determination) or r squared***. ***This denotes the strength of the linear association between x and y. It*** represents the percent of the data that is the closest to the line of best fit.
>
> In essence, (r) tells you about the direction and strength of the relationship, while (R^2) tells you how well the data fits the model (measures the proportion of the variance in the dependent variable that is predictable from the independent variable(s). It ranges from 0 to 1.).
>
> ***Hint5*** It is not appropriate to compute r2 from the nonparametric Spearman correlation coefficient

Example [cats data]: If r = 0.80, then r2 = 0.64, which means that 64% of the total variation in cats heart weight can be explained by the linear relationship between cats body weight and their heat weight. The other 36% of the total variation in heart weight remains unexplained.

r (rs)	Interpertation
1.0	Perfect correlation
0 to 1	The two variables tend to increase or decrease together
0.0	The two variables do not vary together at all
0 to -1	One variable increases as the other decreases
-1.0	Perfect negative or inverse correlation

*** additional details*** some ref give more detailed classification on correlation. for example

(r = 1): Perfect positive correlation (0.7-1): Strong positive correlation. (0.3-0.7): Moderate positive correlation. (0.1-0.3): Weak positive correlation. (r = 0): No correlation. same on the other negative side

> ***Hint6*** r and R2 interpritation alone is not enough to accuratly explain your correlation data. along with these parameters, you need to check the ***p value***. if the p value is significant and you have some correlation pattern, then you are on the right way. in some cases, p value is not significant while you can see a correlation pattern. so be careful.
>
> If you are comparing 2 variables to see their correlation [cats example, or next players example]. you will get the result like this

Players example (correlation between 2 variable)

```
player1 <- c(1,2,3,4,5,6,7,8,9,10)
player2 <- c(3,6,5,4,7,8,9,4,5,6)
# 2 separate sets not in a dataframe as in cats example

cor.test(player1,player2, method="spearman",exact=FALSE)
```

```
##
##  Spearman's rank correlation rho
##
## data:  player1 and player2
## S = 115.55, p-value = 0.4002
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.2997068
```

as you can see, the value of the correlation coefficient is 0.29. moreover, p value is non significant which indicates a non signficant correlation between player 1 snd 2

to get R2, simply raise the r to power of 2

```
0.2997068 ^2 # but its meaningless here in spareman as i said earlier
```

```
## [1] 0.08982417
```

Now you have r, r2 and p

Correlation matrix

If your data have 2 variables or more and you want to correlate between them you need to use cor() function.

Lets use mtcars dataset # datasets package

```
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
```

```
str(mtcars)# to show how your data looks like. 32 observation, 11 variables.
```

```
## '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 ...
```

selecting columns from disp to qsec

```
cars <- mtcars %>% select(disp,hp,drat,wt,qsec)

# read mtcars, and select x variables(columns), then name it cars.


# make sure plyr packge is loaded then dplyr package[of coarse packages needs to be downloaded before l
#library(plyr)
# library(dplyr)
# %>%  "piping or chaining" in dplyr package
```

```
summary(cars)# to see if your data have NA that might spoil your result
```

```
##       disp             hp             drat             wt
##  Min.   : 71.1   Min.   : 52.0   Min.   :2.760   Min.   :1.513
##  1st Qu.:120.8   1st Qu.: 96.5   1st Qu.:3.080   1st Qu.:2.581
##  Median :196.3   Median :123.0   Median :3.695   Median :3.325
##  Mean   :230.7   Mean   :146.7   Mean   :3.597   Mean   :3.217
##  3rd Qu.:326.0   3rd Qu.:180.0   3rd Qu.:3.920   3rd Qu.:3.610
##  Max.   :472.0   Max.   :335.0   Max.   :4.930   Max.   :5.424
##       qsec
##  Min.   :14.50
##  1st Qu.:16.89
##  Median :17.71
##  Mean   :17.85
##  3rd Qu.:18.90
##  Max.   :22.90
```

Lets run the correlation

```
cor1 <- cor(cars)# default is pearson.
print(cor1)# to show result
```

```
##            disp         hp        drat         wt        qsec
## disp  1.0000000  0.7909486 -0.71021393  0.8879799 -0.43369788
## hp    0.7909486  1.0000000 -0.44875912  0.6587479 -0.70822339
## drat -0.7102139 -0.4487591  1.00000000 -0.7124406  0.09120476
## wt    0.8879799  0.6587479 -0.71244065  1.0000000 -0.17471588
## qsec -0.4336979 -0.7082234  0.09120476 -0.1747159  1.00000000
```

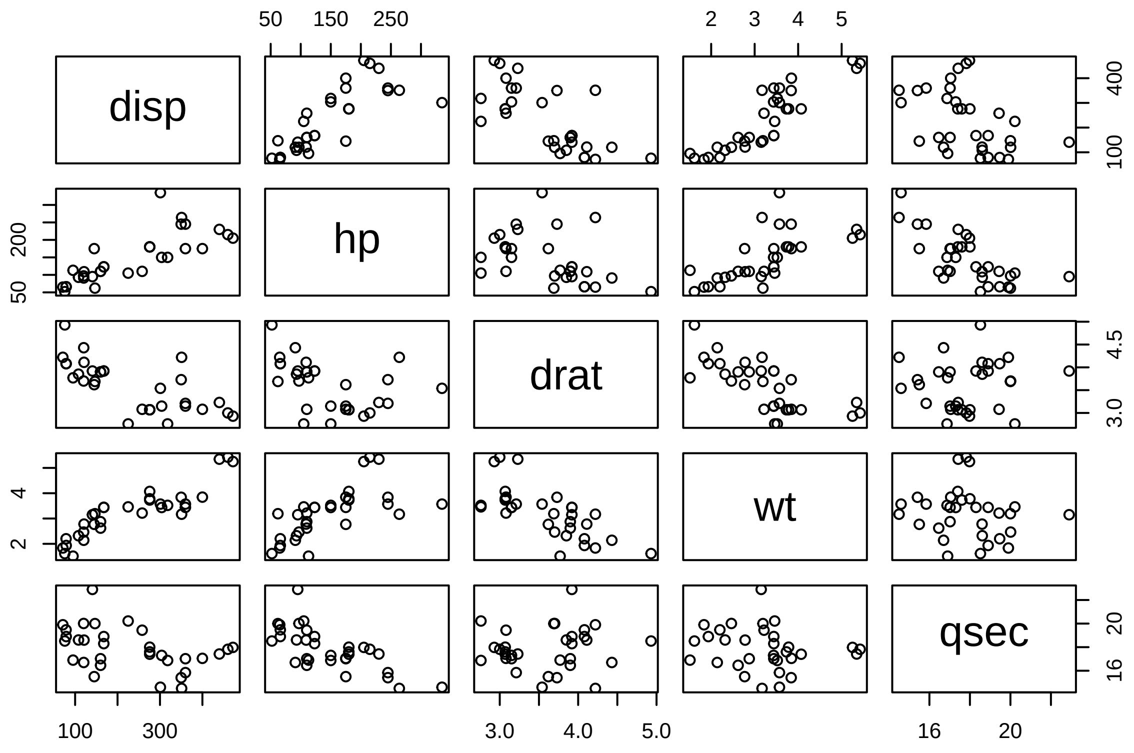
```
cor2 <- cor(cars,method = "spearman")
print(cor2)
```

```
##            disp         hp        drat         wt        qsec
## disp  1.0000000  0.8510426 -0.68359210  0.8977064 -0.45978176
## hp    0.8510426  1.0000000 -0.52012499  0.7746767 -0.66660602
## drat -0.6835921 -0.5201250  1.00000000 -0.7503904  0.09186863
## wt    0.8977064  0.7746767 -0.75039041  1.0000000 -0.22540120
## qsec -0.4597818 -0.6666060  0.09186863 -0.2254012  1.00000000
```

Note: If your data na.rm=TRUE "your data includes missing values or NA", then you need to add *[use = "na.or.complete"]*

then you can draw a simple correlation matrix

```
pairs(cars) # note that i plotted here the original data not the correlation result dataset"cor1"
```

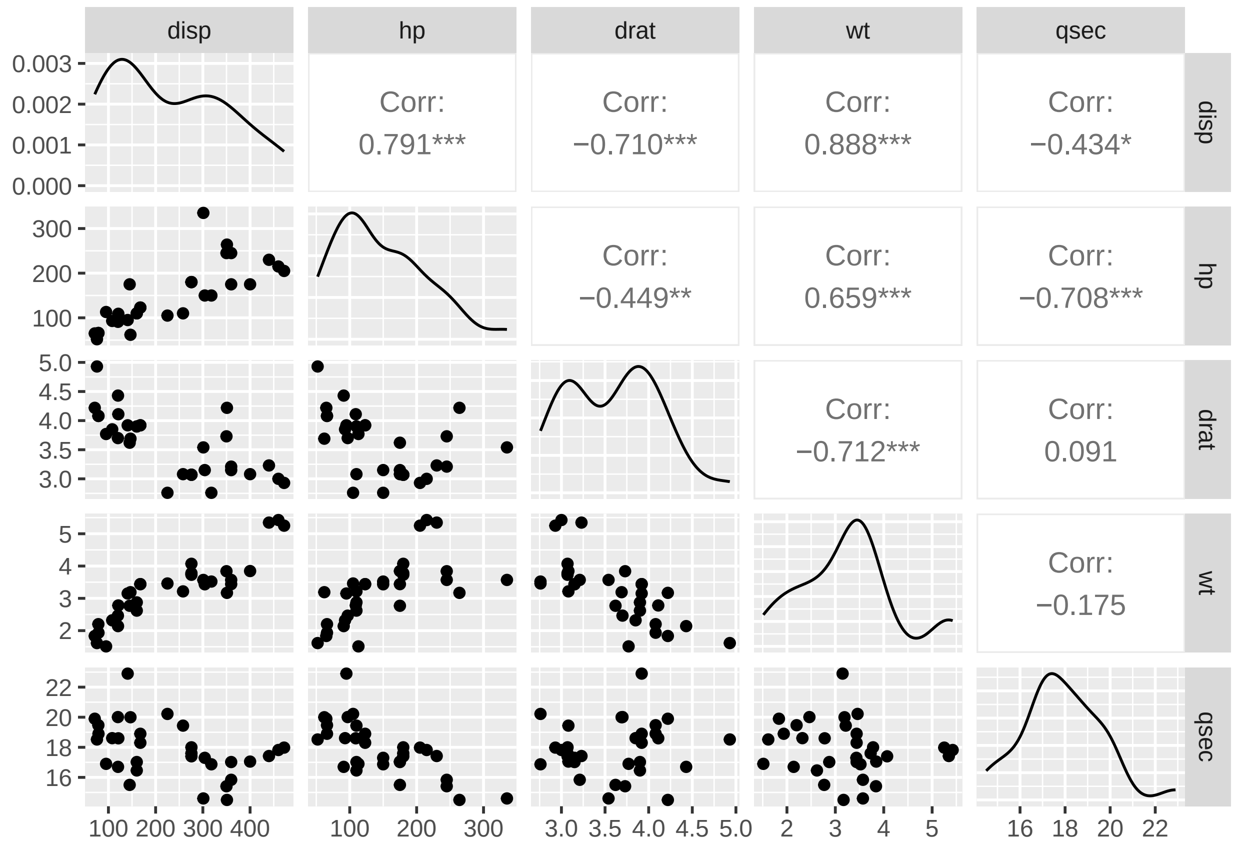


```
# plot(cars) will give the same result)
```

Another package that we can use

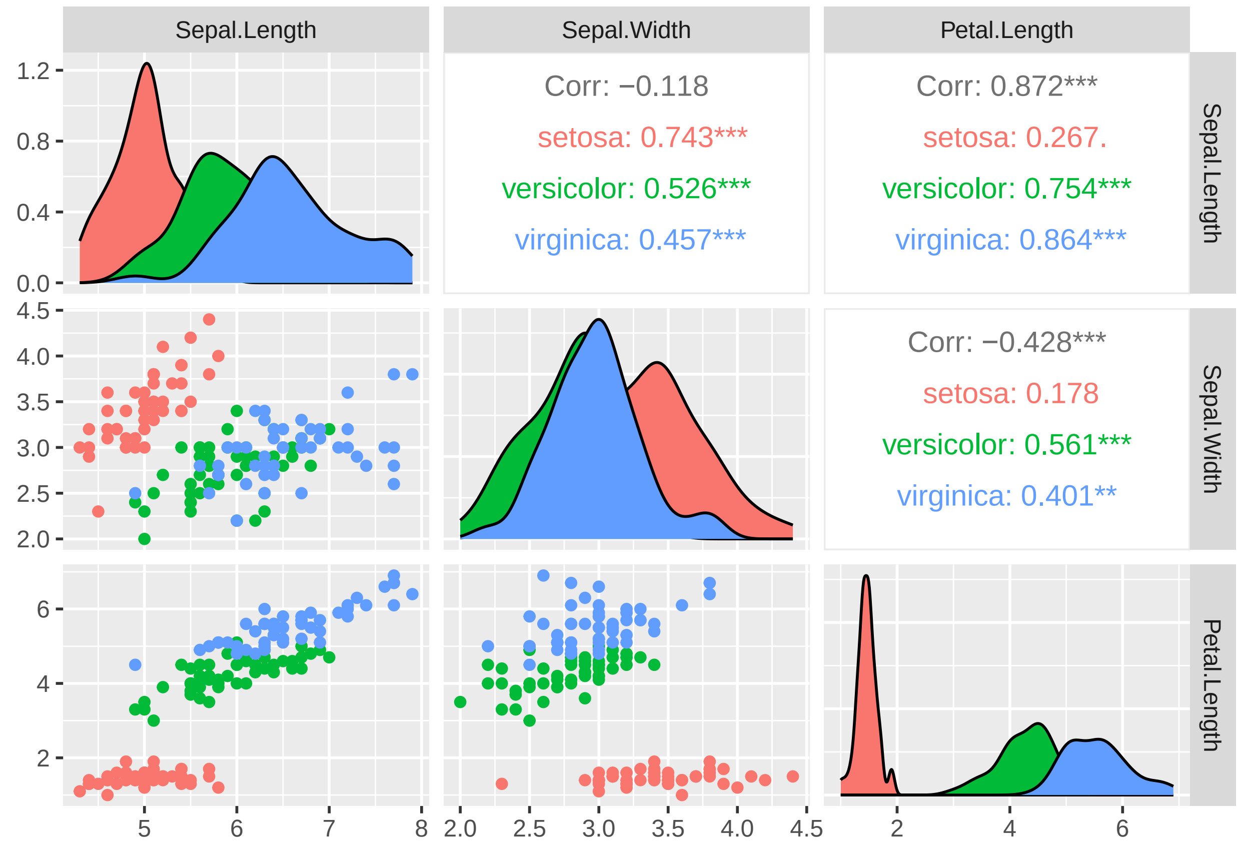
#install GGally package

```
ggpairs(cars)
```



you could specify many things

```
ggpairs(iris, columns = 1:3, aes(colour = Species))
```



using cor function with several variables returns a table includes r. There is no info about R2 or p value

of course r2 can be easily obtained as earlier

```
cor1# this is r
```

```
##            disp         hp        drat         wt        qsec
## disp  1.0000000  0.7909486 -0.71021393  0.8879799 -0.43369788
## hp    0.7909486  1.0000000 -0.44875912  0.6587479 -0.70822339
## drat -0.7102139 -0.4487591  1.00000000 -0.7124406  0.09120476
## wt    0.8879799  0.6587479 -0.71244065  1.0000000 -0.17471588
## qsec -0.4336979 -0.7082234  0.09120476 -0.1747159  1.00000000
```

```
cor1^2 # this is R2
```

```
##           disp        hp        drat         wt        qsec
## disp 1.0000000 0.6255997 0.504403822 0.78850834 0.188093852
## hp   0.6255997 1.0000000 0.201384745 0.43394878 0.501580369
## drat 0.5044038 0.2013847 1.000000000 0.50757168 0.008318308
## wt   0.7885083 0.4339488 0.507571675 1.00000000 0.030525638
## qsec 0.1880939 0.5015804 0.008318308 0.03052564 1.000000000
```

unfourtinatly, we need more steps to calculate p for several variables. but it is not that hard.

We need to generate the correlation probability then ask R to tabulate the data in a handy table.

Load these 2 function to generate the correlation probability and generate a flattensquare matrix. Really, you dont need to understand what is inside these function:)

when calling cor.prob or flattenSquareMatrix, R will do these calculations

***first step*, copy and paste these codes one by one. Make sure you can see them on the upper right panel Environment under functions**

```
# these 2 codes can be found in the downloadable material sent to you earlier
# to generate correlation probability

cor.prob <- function (X, dfr = nrow(X) - 2) {
  R <- cor(X, use="pairwise.complete.obs")
  above <- row(R) < col(R)
  r2 <- R[above]^2
  Fstat <- r2 * dfr/(1 - r2)
  R[above] <- 1 - pf(Fstat, 1, dfr)
  R[row(R) == col(R)] <- NA
  R
}
```

```
# ask R to generate readable table

flattenSquareMatrix <- function(m) {
  if (!is.matrix(m) || nrow(m) != ncol(m)) {
    stop("Input must be a square matrix.")
  }
  if (!identical(rownames(m), colnames(m))) {
    stop("Row and column names must match.")
  }
  ut <- upper.tri(m)
  data.frame(
    i = rownames(m)[row(m)[ut]],
    j = rownames(m)[col(m)[ut]],
    cor = m[ut],
    p = m[ut]
  )
}
```

***** second step*** lets get the cor of cars data**

```
head(cars)
```

```
##                    disp  hp drat    wt  qsec
## Mazda RX4           160 110 3.90 2.620 16.46
## Mazda RX4 Wag      160 110 3.90 2.875 17.02
## Datsun 710         108  93 3.85 2.320 18.61
## Hornet 4 Drive     258 110 3.08 3.215 19.44
## Hornet Sportabout  360 175 3.15 3.440 17.02
## Valiant            225 105 2.76 3.460 20.22
```

```
cor(cars)
```

```
##            disp         hp        drat         wt        qsec
## disp  1.0000000  0.7909486 -0.71021393  0.8879799 -0.43369788
## hp    0.7909486  1.0000000 -0.44875912  0.6587479 -0.70822339
## drat -0.7102139 -0.4487591  1.00000000 -0.7124406  0.09120476
## wt    0.8879799  0.6587479 -0.71244065  1.0000000 -0.17471588
## qsec -0.4336979 -0.7082234  0.09120476 -0.1747159  1.00000000
```

***** Third step***, lets use the first function to get p value**

```
cor.prob(cars)
```

```
##            disp            hp          drat            wt         qsec
## disp         NA  7.142679e-08  5.282022e-06  1.222322e-11 1.314404e-02
## hp    0.7909486            NA  9.988772e-03  4.145827e-05 5.766253e-06
## drat -0.7102139 -4.487591e-01            NA  4.784260e-06 6.195826e-01
## wt    0.8879799  6.587479e-01 -7.124406e-01            NA 3.388683e-01
## qsec -0.4336979 -7.082234e-01  9.120476e-02 -1.747159e-01           NA
```

***** Forth step***, lets ask R to tabulate the result in a handy way using the 2nd function**

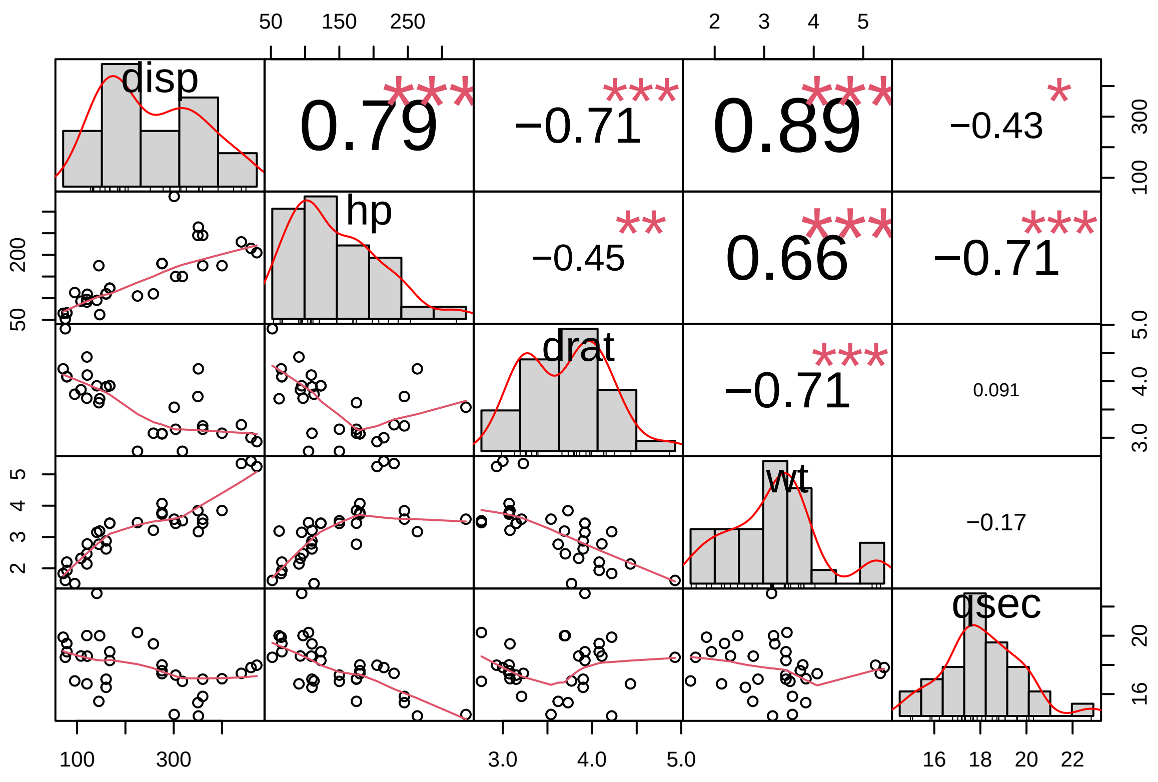
```
flattenSquareMatrix(cor.prob(cars))
```

```
##       i    j          cor            p
## 1  disp   hp 7.142679e-08 7.142679e-08
## 2  disp drat 5.282022e-06 5.282022e-06
## 3    hp drat 9.988772e-03 9.988772e-03
## 4  disp   wt 1.222322e-11 1.222322e-11
## 5    hp   wt 4.145827e-05 4.145827e-05
## 6  drat   wt 4.784260e-06 4.784260e-06
## 7  disp qsec 1.314404e-02 1.314404e-02
## 8    hp qsec 5.766253e-06 5.766253e-06
## 9  drat qsec 6.195826e-01 6.195826e-01
## 10   wt qsec 3.388683e-01 3.388683e-01
```

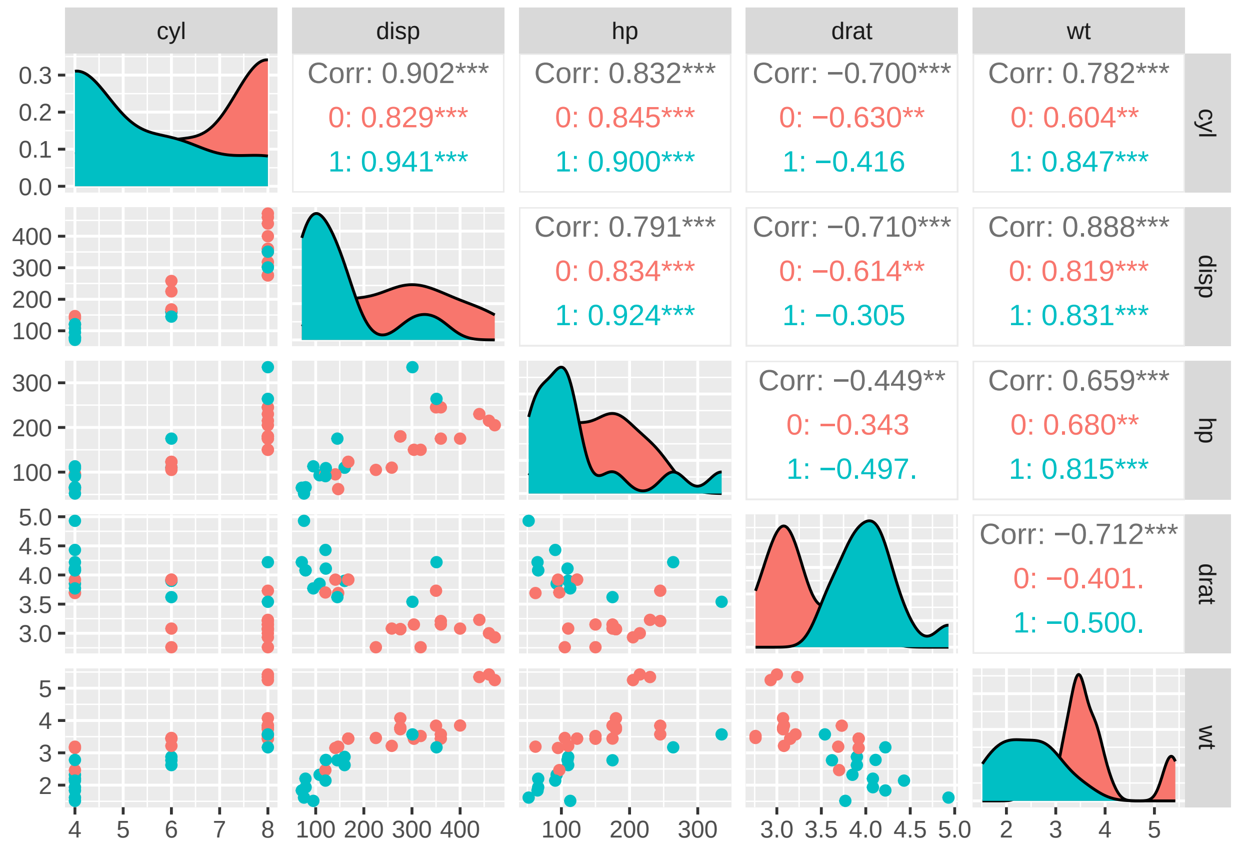
Now you have a table i and j is your variables, cor is r and p is of course p!

> Bonus professional point: graph it [PerformanceAnalytics package is needed] or GGally (extention of ggplot2) as mentioned earlier. # More graphical tools in graphics module courses.

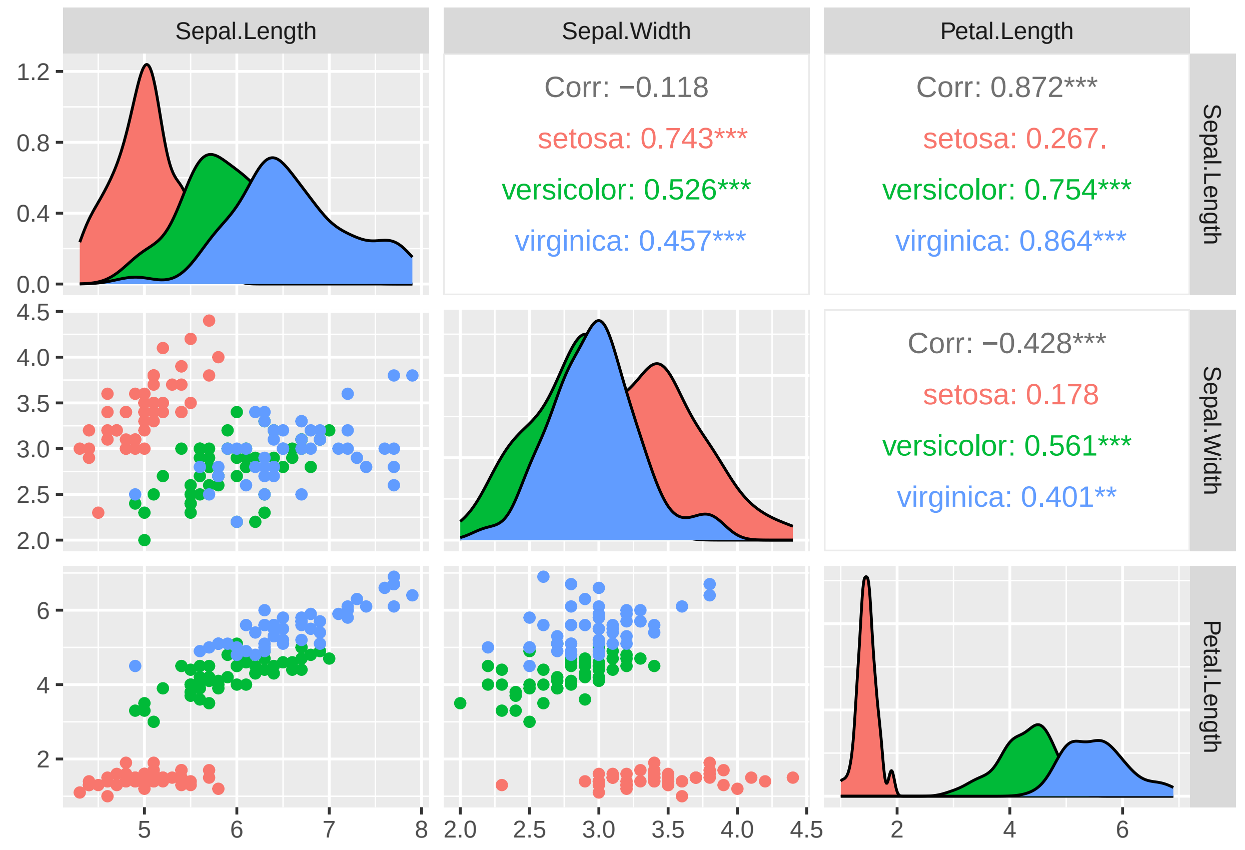
```r
chart.Correlation(cars) # default is pearson
```



```r
ggpairs(mtcars, columns = 2:6, aes(colour = factor(am)))
```



```
ggpairs(iris, columns = 1:3, aes(colour = Species))
```

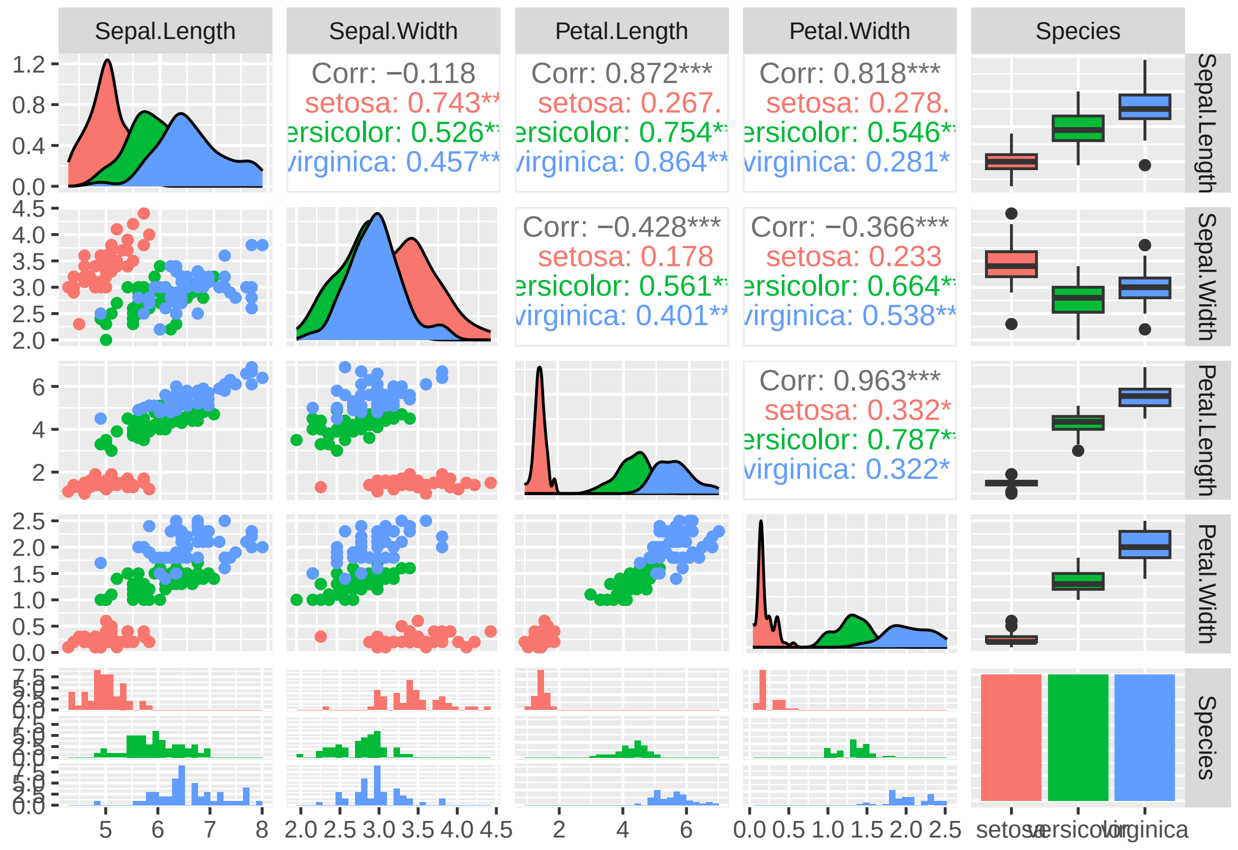


for mtcars, i had to tell R to consider am as factor (not numbers), but i did not do that in the iris data.

compare previous 2 codes.

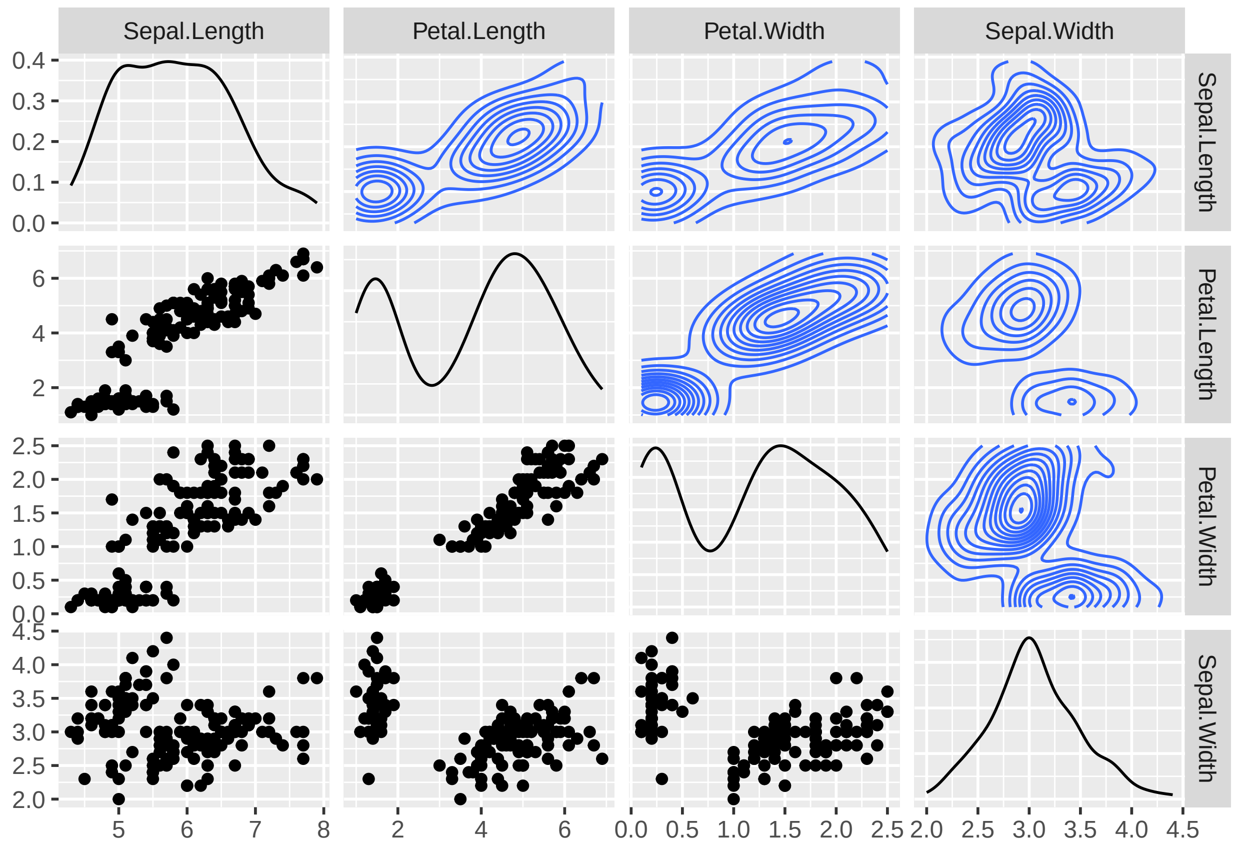
```r
ggpairs(iris, aes(colour = Species))
```

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



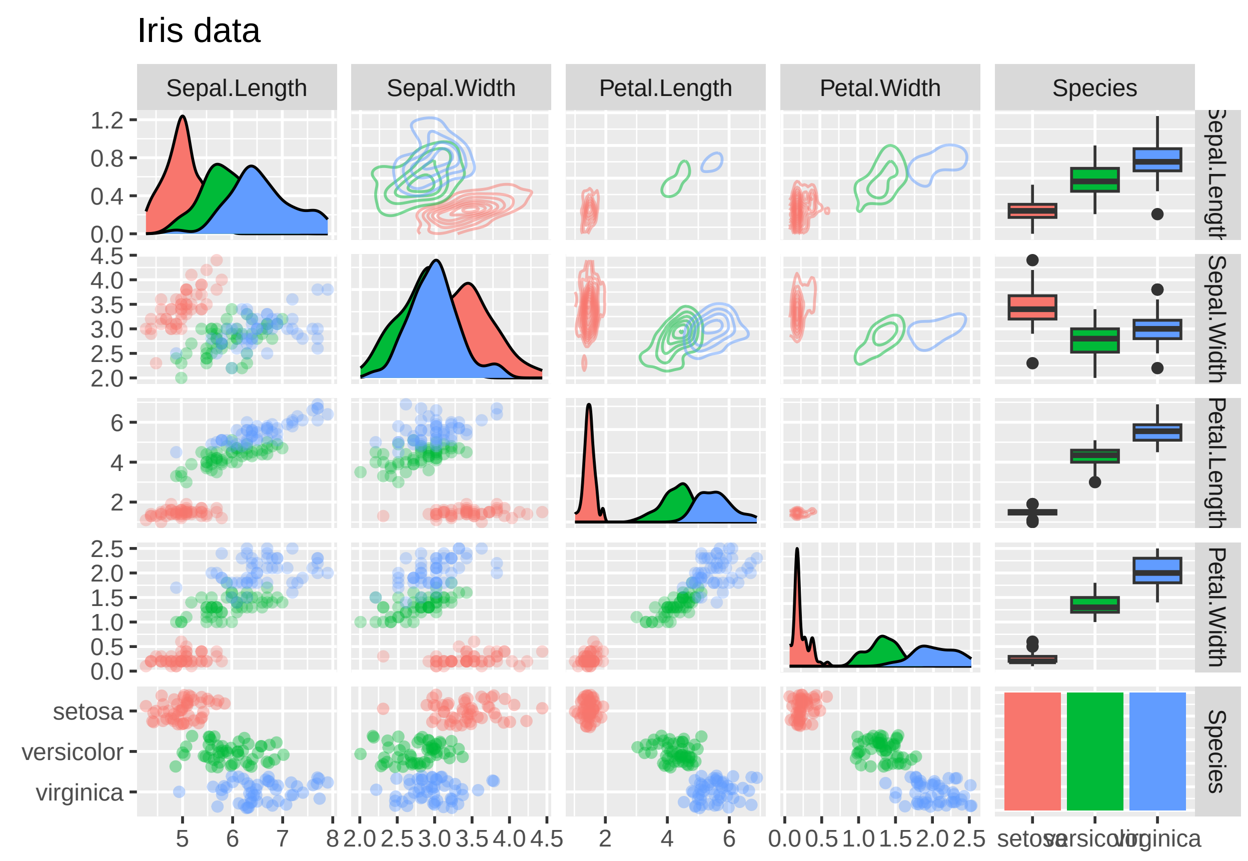
#ggpairs have several layouts, ill put couple of them

```r
ggpairs(
  iris[, c(1, 3, 4, 2)],
  upper = list(continuous = "density", combo = "box_no_facet"),
  lower = list(continuous = "points", combo = "dot_no_facet")
) # note that i selected and rearranged variables here
```



```r
ggpairs(
  iris[, 1:5],
  mapping = ggplot2::aes(color = Species),
  upper = list(continuous = wrap("density", alpha = 0.5), combo = "box_no_facet"),
  lower = list(continuous = wrap("points", alpha = 0.3), combo = wrap("dot_no_facet", alpha = 0.4)),
  title = "Iris data"
)
```

```
## Warning: ‘stat_contour()‘: Zero contours were generated

## Warning in min(x): no non-missing arguments to min; returning Inf

## Warning in max(x): no non-missing arguments to max; returning -Inf

## Warning: ‘stat_contour()‘: Zero contours were generated

## Warning in min(x): no non-missing arguments to min; returning Inf

## Warning in max(x): no non-missing arguments to max; returning -Inf
```



#End of session