Correlation tests and correlation matrix in R

Igor Hut

11 January, 2017

Contents

The following content is mostly based on the material that can be found at http://www.sthda.com/, as well as in the vignette for `corrplot` R package - An Introduction to corrplot Package. Since I'm a huge fan of Hadley Wickham's work I'll insist on solutions based in "tidyverse" whenewer possible...

Install and load required R packages

We'll use the `ggpubr` R package for an easy ggplot2-based data visualization, `corrplot` package to plot correlograms, `Hmisc` to calculate correlation matrices containing both cor. coefs. and p-values, and of course `tidyverse` for all the data wrangling, plotting and alike:

```
library(ggpubr)
library(tidyverse)
library(Hmisc)
```

Methods for correlation analyses

There are different methods to perform correlation analysis:

- **Pearson correlation (r)**, which measures a linear dependence between two variables (x and y). It's also known as a parametric correlation test because it depends to the distribution of the data. It can be used only when x and y are from normal distribution. The plot of $y = f(x)$ is named the *linear regression curve.*
- **Kendall** τ and **Spearman** ρ, which are rank-based correlation coefficients (non-parametric)
- **The most commonly used method is the Pearson correlation method**

Compute correlation in R

R functions

Correlation coefficients can be computed in R by using the functions `cor()` and `cor.test()`:

- `cor()` computes the correlation coefficient
- `cor.test()` test for association/correlation between paired samples. It returns both the correlation coefficient and the significance level(or p-value) of the correlation.

The simplified formats are:

```
cor(x, y, method = c("pearson", "kendall", "spearman"))
cor.test(x, y, method=c("pearson", "kendall", "spearman"))
```

where:

- x, y: numeric vectors with the same length
- method: correlation method

If the data contain missing values, the following R code can be used to handle missing values by case-wise deletion:

```
cor(x, y,  method = "pearson", use = "complete.obs")
```

Preliminary considerations

We'll use the well known built-in `mtcars` R dataset.

```
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
```

We'd like to compute the correlation between `mpg` and `wt` variables.

First let's visualise our data by the means of a scatter plot. We'll be using `ggpubr` R package

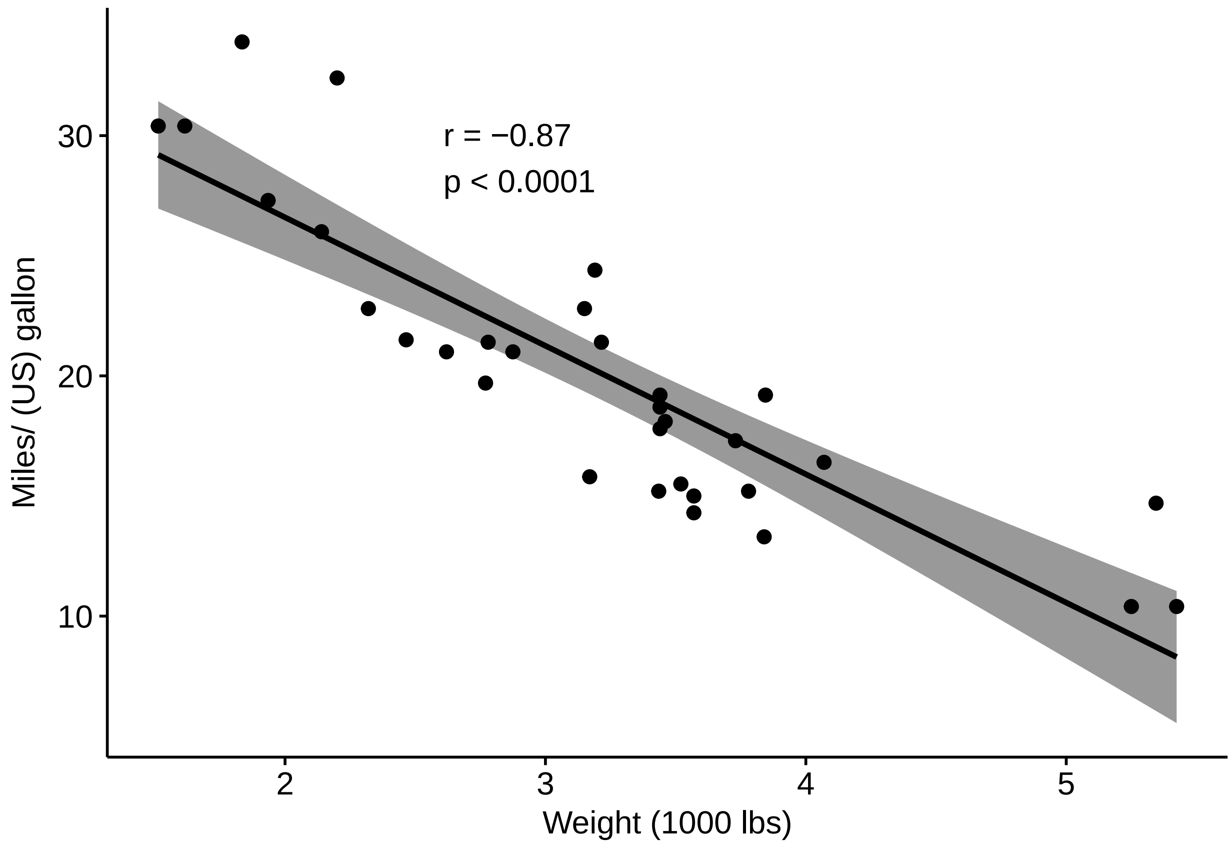
```
library(ggpubr)

my_data <- mtcars
my_data$cyl <- factor(my_data$cyl)
str(my_data)
```

```
## '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 : Factor w/ 3 levels "4","6","8": 2 2 1 2 3 2 3 1 1 2 ...
##  $ 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 ...
```

```
ggscatter(my_data, x = "wt", y = "mpg",
          add = "reg.line", conf.int = TRUE,
          cor.coef = TRUE, cor.method = "pearson",
          xlab = "Weight (1000 lbs)", ylab = "Miles/ (US) gallon")
```



Preleminary test to check the test assumptions

1. Is the relation between variables linear? Yes, from the plot above, the relationship can be, closely enough, modeled as linear. In the situation where the scatter plots show curved patterns, we are dealing with nonlinear association between the two variables.
2. Are the data from each of the 2 variables (x, y) following a normal distribution?
   - Use *Shapiro-Wilk* normality test -> R function: `shapiro.test()`
   - and look at the normality plot -> R function: `ggpubr::ggqqplot()`

- *Shapiro-Wilk* test can be performed as follow:
  - Null hypothesis: the data are normally distributed
  - Alternative hypothesis: the data are not normally distributed

```
# Shapiro-Wilk normality test for mpg
shapiro.test(my_data$mpg) # => p = 0.1229
```

```
##
##  Shapiro-Wilk normality test
```

```
##
## data:  my_data$mpg
## W = 0.94756, p-value = 0.1229
```

```r
# Shapiro-Wilk normality test for wt
shapiro.test(my_data$wt) # => p = 0.09
```

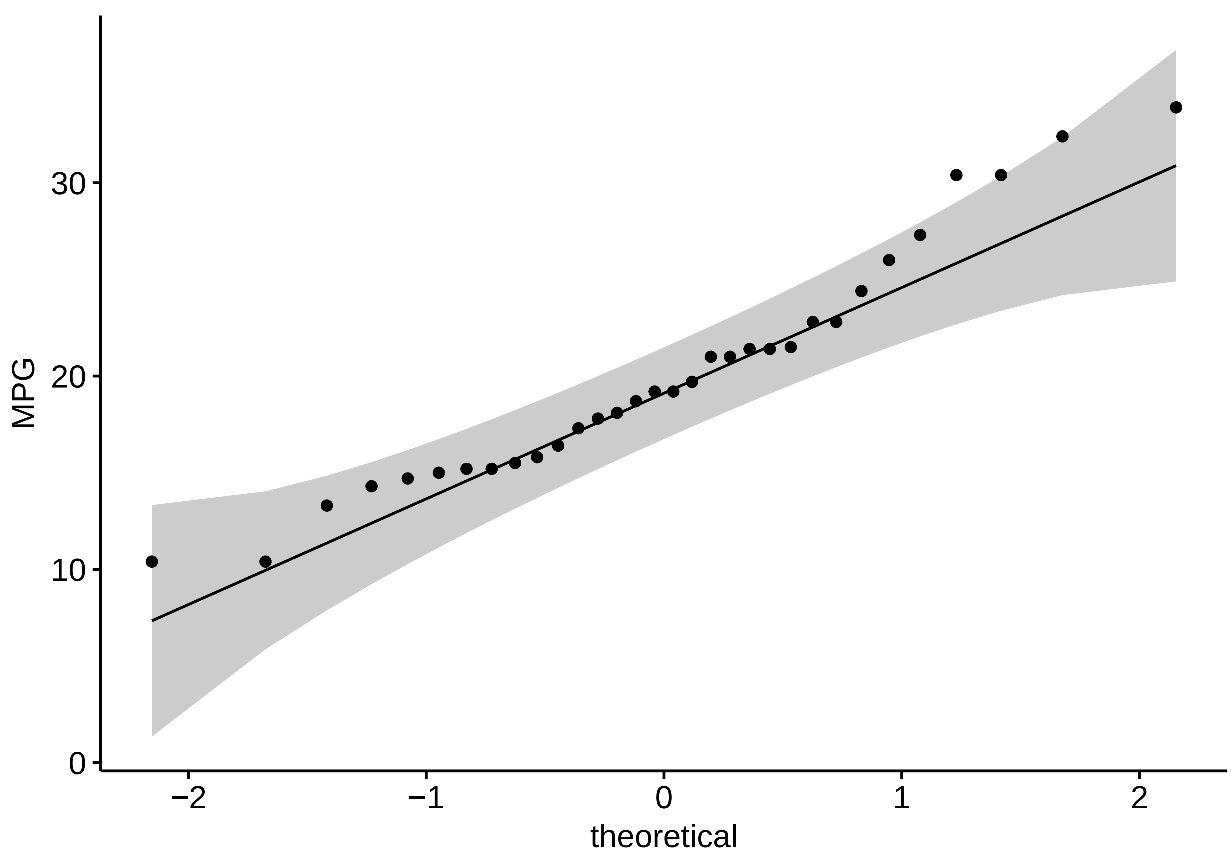
```
##
##  Shapiro-Wilk normality test
##
## data:  my_data$wt
## W = 0.94326, p-value = 0.09265
```

As can be seen from the output, the two p-values are greater than the predetermined significance level of 0.05 implying that the distribution of the data are not significantly different from normal distribution. In other words, we can assume the normality.

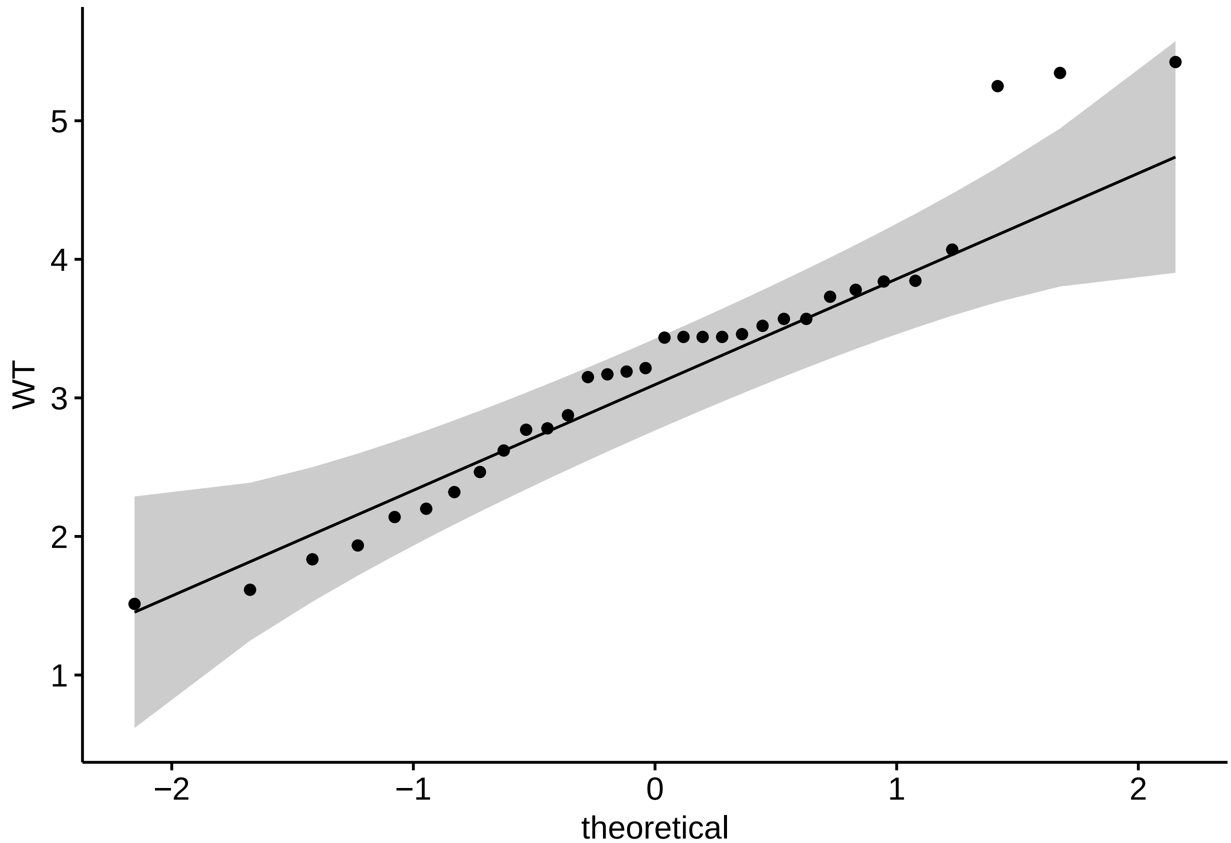
- One more option for checking the normality of the data distribution is visual inspection of the Q-Q plots (quantile-quantile plots). Q-Q plot draws the correlation between a given sample and the theoretical normal distribution.

Again, we'll use the `ggpubr` R package to obtain "pretty", i.e. publishing-ready, Q-Q plots.

```r
library("ggpubr")
# Check for the normality of "mpg""
ggqqplot(my_data$mpg, ylab = "MPG")
```




```r
# Check for the normality of "wt""
ggqqplot(my_data$wt, ylab = "WT")
```



From the Q-Q normality plots, we can assume that both samples may come from populations that, closely enough, follow normal distributions.

It is important to note that if the data does not follow the normal distribution, at least closely enough, it's recommended to use the non-parametric correlation, including Spearman and Kendall rank-based correlation tests.

Pearson correlation test

Example:

```
res <- cor.test(my_data$wt, my_data$mpg, method = "pearson")
res
```

```
##
##  Pearson's product-moment correlation
##
## data:  my_data$wt and my_data$mpg
## t = -9.559, df = 30, p-value = 1.294e-10
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.9338264 -0.7440872
## sample estimates:
##        cor
## -0.8676594
```

So what's happening here? First of all let's clarify the meaning of this printout:

- `t` is the *t-test* statistic value (`t = -9.559`),
- `df` is the degrees of freedom (`df= 30`),
- `p-value` is the significance level of the *t-test* (`p-value =` 1.29410^{-10}).
- `conf.int` is the *confidence interval* of the correlation coefficient at 95% (`conf.int = [-0.9338, -0.7441]`);
- `sample estimates` is the *correlation coefficient* (`Cor.coeff = -0.87`).

Interpretation of the results: As can be see from the results above the *p-value* of the test is 1.29410^{-10}, which is less than the significance level $\alpha = 0.05$. We can conclude that `wt` and `mpg` are significantly correlated with a correlation coefficient of -0.87 and *p-value* of 1.29410^{-10}.

Access to the values returned by `cor.test()` function

The function `cor.test()` returns a list containing the following components:

```
str(res)
```

```
## List of 9
##  $ statistic  : Named num -9.56
##   ..- attr(*, "names")= chr "t"
##  $ parameter  : Named int 30
##   ..- attr(*, "names")= chr "df"
##  $ p.value    : num 1.29e-10
##  $ estimate   : Named num -0.868
##   ..- attr(*, "names")= chr "cor"
##  $ null.value : Named num 0
##   ..- attr(*, "names")= chr "correlation"
##  $ alternative: chr "two.sided"
##  $ method     : chr "Pearson's product-moment correlation"
##  $ data.name  : chr "my_data$wt and my_data$mpg"
##  $ conf.int   : atomic [1:2] -0.934 -0.744
##   ..- attr(*, "conf.level")= num 0.95
##  - attr(*, "class")= chr "htest"
```

Of these we are most interested with:

- `p.value`: the p-value of the test
- `estimate`: the correlation coefficient

```
# Extract the p.value
res$p.value
```

```
## [1] 1.293959e-10
```

```
# Extract the correlation coefficient
res$estimate
```

```
##        cor
## -0.8676594
```

Kendall rank correlation test

The Kendall rank correlation coefficient or **Kendall's τ statistic** is used to estimate a rank-based measure of association. This test may be used if the data do not necessarily come from a bivariate normal distribution.

```
res2 <- cor.test(my_data$mpg, my_data$wt, method = "kendall")
```

```
## Warning in cor.test.default(my_data$mpg, my_data$wt, method = "kendall"):
```

```
## Cannot compute exact p-value with ties
```

```r
res2
```

```
##
##  Kendall's rank correlation tau
##
## data:  my_data$mpg and my_data$wt
## z = -5.7981, p-value = 6.706e-09
## alternative hypothesis: true tau is not equal to 0
## sample estimates:
##        tau
## -0.7278321
```

Here `tau` is the Kendall correlation coefficient, so The correlation coefficient between `mpg` and `wy` is -0.7278 and the p-value is 6.70610^{-9}.

Spearman rank correlation coefficient

Spearman's ρ statistic is also used to estimate a rank-based measure of association. This test may be used if the data do not come from a bivariate normal distribution.

```r
res3 <- cor.test(my_data$wt, my_data$mpg, method = "spearman")
```

```
## Warning in cor.test.default(my_data$wt, my_data$mpg, method = "spearman"):
## Cannot compute exact p-value with ties
```

```r
res3
```

```
##
##  Spearman's rank correlation rho
##
## data:  my_data$wt and my_data$mpg
## S = 10292, p-value = 1.488e-11
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## -0.886422
```

Here, `rho` is the Spearman's correlation coefficient, so the correlation coefficient between `mpg` and `wt` is -0.8864 and the p-value is 1.48810^{-11}.

How to interpret correlation coefficient

Value of the correlation coefficient can vary between -1 and 1:

- -1 indicates a strong negative correlation : this means that every time x increases, y decreases
- 0 means that there is no association between the two variables (x and y)
- 1 indicates a strong positive correlation : this means that y increases with x

What is a correlation matrix?

Previously, we described how to perform correlation test between two variables. In the following sections we'll see how a **correlation matrix** can be computed and visualized. The **correlation matrix** is used to investigate the dependence between multiple variables at the same time. The result is a table containing the correlation coefficients between each variable and the others.

Compute correlation matrix in R

We have already mentioned the `cor()` function, at the intoductory part of this document dealing with the correlation test for a bivariate case. It be used to compute a correlation matrix. A simplified format of the function is :

```
cor(x, method = c("pearson", "kendall", "spearman"))
```

Here:

- `x` is numeric matrix or a data frame.
- `method`: indicates the correlation coefficient to be computed. The default is "pearson"" correlation coefficient which measures the linear dependence between two variables. As already explained "kendall" and "spearman" correlation methods are non-parametric rank-based correlation tests.

If your data contain missing values, the following R code can be used to handle missing values by case-wise deletion:

```
cor(x, method = "pearson", use = "complete.obs")
```

Plain correlation matrix

Example:

```
library(dplyr)

my_data <- select(mtcars, mpg, disp, hp, drat, wt, qsec)
head(my_data)
```

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

```
#Let's compute the correlation matrix
cor_1 <- round(cor(my_data), 2)
cor_1
```

```
##        mpg  disp    hp  drat    wt  qsec
## mpg   1.00 -0.85 -0.78  0.68 -0.87  0.42
## disp -0.85  1.00  0.79 -0.71  0.89 -0.43
## hp   -0.78  0.79  1.00 -0.45  0.66 -0.71
## drat  0.68 -0.71 -0.45  1.00 -0.71  0.09
## wt   -0.87  0.89  0.66 -0.71  1.00 -0.17
## qsec  0.42 -0.43 -0.71  0.09 -0.17  1.00
```

Unfortunately, the function `cor()` returns only the correlation coefficients between variables. In the next section, we will use `Hmisc` R package to calculate the correlation *p-values*.

Correlation matrix with significance levels (p-value)

The function `rcorr()` (in `Hmisc` package) can be used to compute the significance levels for pearson and spearman correlations. It returns both the *correlation coefficients* and the *p-value* of the correlation for all possible pairs of columns in the data table.

Simplified format:

```
rcorr(x, type = c("pearson","spearman"))
```

x should be a matrix. The correlation type can be either *pearson* or *spearman*.

Example:

```
library("Hmisc")

cor_2 <- rcorr(as.matrix(my_data))
cor_2
```

```
##        mpg  disp    hp  drat    wt  qsec
## mpg   1.00 -0.85 -0.78  0.68 -0.87  0.42
## disp -0.85  1.00  0.79 -0.71  0.89 -0.43
## hp   -0.78  0.79  1.00 -0.45  0.66 -0.71
## drat  0.68 -0.71 -0.45  1.00 -0.71  0.09
## wt   -0.87  0.89  0.66 -0.71  1.00 -0.17
## qsec  0.42 -0.43 -0.71  0.09 -0.17  1.00
##
## n= 32
##
##
## P
##      mpg    disp   hp     drat   wt     qsec
## mpg         0.0000 0.0000 0.0000 0.0000 0.0171
## disp 0.0000        0.0000 0.0000 0.0000 0.0131
## hp   0.0000 0.0000        0.0100 0.0000 0.0000
## drat 0.0000 0.0000 0.0100        0.0000 0.6196
## wt   0.0000 0.0000 0.0000 0.0000        0.3389
## qsec 0.0171 0.0131 0.0000 0.6196 0.3389
```

The output of the function `rcorr()` is a list containing the following elements :

- **r** : the correlation matrix
- **n** : the matrix of the number of observations used in analyzing each pair of variables
- **P** : the p-values corresponding to the significance levels of correlations.

Extracting the p-values or the correlation coefficients from the output:

```
str(cor_2)
```

```
## List of 3
##  $ r: num [1:6, 1:6] 1 -0.848 -0.776 0.681 -0.868 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:6] "mpg" "disp" "hp" "drat" ...
##   .. ..$ : chr [1:6] "mpg" "disp" "hp" "drat" ...
##  $ n: int [1:6, 1:6] 32 32 32 32 32 32 32 32 32 32 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:6] "mpg" "disp" "hp" "drat" ...
##   .. ..$ : chr [1:6] "mpg" "disp" "hp" "drat" ...
##  $ P: num [1:6, 1:6] NA 9.38e-10 1.79e-07 1.78e-05 1.29e-10 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:6] "mpg" "disp" "hp" "drat" ...
##   .. ..$ : chr [1:6] "mpg" "disp" "hp" "drat" ...
##  - attr(*, "class")= chr "rcorr"
```

```r
# As you can see "cor_2" is a list so extracting these values is quite simple...

# p-values
cor_2$P
```

```
##               mpg         disp           hp         drat           wt
## mpg            NA 9.380354e-10 1.787838e-07 1.776241e-05 1.293956e-10
## disp 9.380354e-10           NA 7.142686e-08 5.282028e-06 1.222311e-11
## hp   1.787838e-07 7.142686e-08           NA 9.988768e-03 4.145833e-05
## drat 1.776241e-05 5.282028e-06 9.988768e-03           NA 4.784268e-06
## wt   1.293956e-10 1.222311e-11 4.145833e-05 4.784268e-06           NA
## qsec 1.708199e-02 1.314403e-02 5.766250e-06 6.195823e-01 3.388682e-01
##              qsec
## mpg  1.708199e-02
## disp 1.314403e-02
## hp   5.766250e-06
## drat 6.195823e-01
## wt   3.388682e-01
## qsec           NA
```

```r
# Correlation matrix
cor_2$r
```

```
##             mpg       disp         hp        drat         wt        qsec
## mpg   1.0000000 -0.8475513 -0.7761683  0.68117189 -0.8676594  0.41868404
## disp -0.8475513  1.0000000  0.7909486 -0.71021390  0.8879799 -0.43369791
## hp   -0.7761683  0.7909486  1.0000000 -0.44875914  0.6587479 -0.70822340
## drat  0.6811719 -0.7102139 -0.4487591  1.00000000 -0.7124406  0.09120482
## wt   -0.8676594  0.8879799  0.6587479 -0.71244061  1.0000000 -0.17471591
## qsec  0.4186840 -0.4336979 -0.7082234  0.09120482 -0.1747159  1.00000000
```

Custom function for convinient formatting of the correlation matrix

This section provides a simple function for formatting a correlation matrix into a table with 4 columns containing :

- Column 1 : row names (variable 1 for the correlation test)
- Column 2 : column names (variable 2 for the correlation test)
- Column 3 : the correlation coefficients
- Column 4 : the p-values of the correlations

```r
flat_cor_mat <- function(cor_r, cor_p){
  #This function provides a simple formatting of a correlation matrix
  #into a table with 4 columns containing :
    # Column 1 : row names (variable 1 for the correlation test)
    # Column 2 : column names (variable 2 for the correlation test)
    # Column 3 : the correlation coefficients
    # Column 4 : the p-values of the correlations
  library(tidyr)
  library(tibble)
  cor_r <- rownames_to_column(as.data.frame(cor_r), var = "row")
  cor_r <- gather(cor_r, column, cor, -1)
  cor_p <- rownames_to_column(as.data.frame(cor_p), var = "row")
  cor_p <- gather(cor_p, column, p, -1)
  cor_p_matrix <- left_join(cor_r, cor_p, by = c("row", "column"))
```

```r
  cor_p_matrix
}

cor_3 <- rcorr(as.matrix(mtcars[, 1:7]))

my_cor_matrix <- flat_cor_mat(cor_3$r, cor_3$P)
head(my_cor_matrix)
```

```
##    row column        cor            p
## 1  mpg    mpg  1.0000000           NA
## 2  cyl    mpg -0.8521619 6.112697e-10
## 3 disp    mpg -0.8475513 9.380354e-10
## 4   hp    mpg -0.7761683 1.787838e-07
## 5 drat    mpg  0.6811719 1.776241e-05
## 6   wt    mpg -0.8676594 1.293956e-10
```

Visualization of a correlation matrix