Corrplot: A Powerful Tool for Visualizing Correlation Matrices

Emma Wang

November 27, 2017

Introduction

This post will introduce corrplot, a R package used for exploratory data analysis, that generates a graphical display of a correlation matrix. It visualizes the dependence between all possible pairs of variables at the same time. This is very useful in the beginning when we just started exploring a data set, as it gives a clear idea of which variables have the strongest relationships. This post will demonstrate the use and meaning of correlation matrix and corrplot, with an emphasis on the aesthetic elements.

Correlation Matrix

Before I dive deep into corrplot, let's first talk about what a correlation matrix really is. Let's generate a plain correlation matrix with the function `cor()`, using the built-in data set `mtcars`.

Take a look at the data:

```
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
```

```
cor(mtcars)
```

```
##             mpg        cyl       disp         hp        drat         wt
## mpg   1.0000000 -0.8521620 -0.8475514 -0.7761684  0.68117191 -0.8676594
## cyl  -0.8521620  1.0000000  0.9020329  0.8324475 -0.69993811  0.7824958
## disp -0.8475514  0.9020329  1.0000000  0.7909486 -0.71021393  0.8879799
## hp   -0.7761684  0.8324475  0.7909486  1.0000000 -0.44875912  0.6587479
## drat  0.6811719 -0.6999381 -0.7102139 -0.4487591  1.00000000 -0.7124406
## wt   -0.8676594  0.7824958  0.8879799  0.6587479 -0.71244065  1.0000000
## qsec  0.4186840 -0.5912421 -0.4336979 -0.7082234  0.09120476 -0.1747159
## vs    0.6640389 -0.8108118 -0.7104159 -0.7230967  0.44027846 -0.5549157
## am    0.5998324 -0.5226070 -0.5912270 -0.2432043  0.71271113 -0.6924953
## gear  0.4802848 -0.4926866 -0.5555692 -0.1257043  0.69961013 -0.5832870
## carb -0.5509251  0.5269883  0.3949769  0.7498125 -0.09078980  0.4276059
##             qsec         vs          am       gear        carb
## mpg   0.41868403  0.6640389  0.59983243  0.4802848 -0.55092507
## cyl  -0.59124207 -0.8108118 -0.52260705 -0.4926866  0.52698829
## disp -0.43369788 -0.7104159 -0.59122704 -0.5555692  0.39497686
## hp   -0.70822339 -0.7230967 -0.24320426 -0.1257043  0.74981247
## drat  0.09120476  0.4402785  0.71271113  0.6996101 -0.09078980
## wt   -0.17471588 -0.5549157 -0.69249526 -0.5832870  0.42760594
## qsec  1.00000000  0.7445354 -0.22986086 -0.2126822 -0.65624923
## vs    0.74453544  1.0000000  0.16834512  0.2060233 -0.56960714
## am   -0.22986086  0.1683451  1.00000000  0.7940588  0.05753435
## gear -0.21268223  0.2060233  0.79405876  1.0000000  0.27407284
## carb -0.65624923 -0.5696071  0.05753435  0.2740728  1.00000000
```

As seen above, choosing a data frame as the argument of `cor()` returns a correlation matrix, which displays the correlations between each possible pair of variables. However, the problem here is that it is very easy to look past certain correlation values that may be meaningful, because we are jammed with too many numbers. The corrplot package deals with this issue.

Corrplot()

(Install and) load the package:

```
library(corrplot)
```

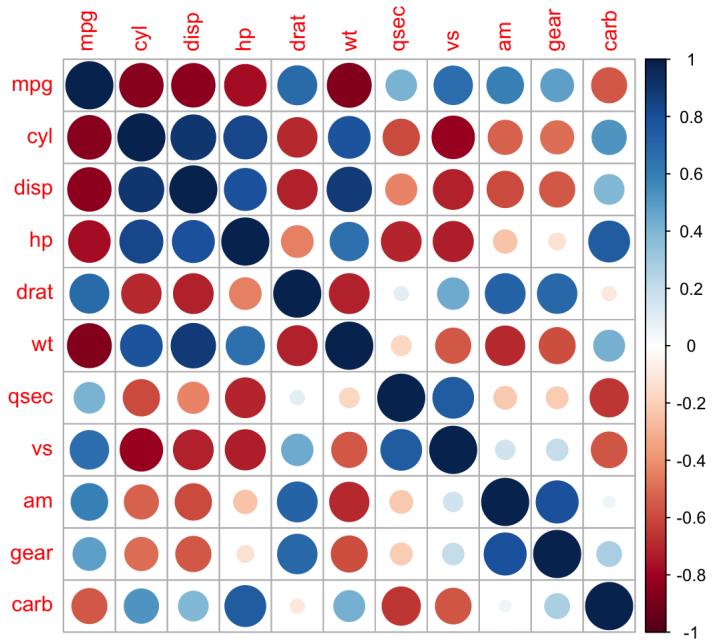
```
## Warning: package 'corrplot' was built under R version 3.4.2
```

```
## corrplot 0.84 loaded
```

Basic Example

With the same dataset `mtcars`, let's see how `corrplot()` visualizes it, starting with the most basic example. The graphical display of a correlation matrix is called a **correlogram**. It is important to note that `corrplot` only takes in matrix as argument, not dataframe.

```
## The corrplot package cannot take in arguments as data frames, so we need to convert mtcars into its correlation matrix.
plotted = cor(mtcars)
corrplot(plotted)
```

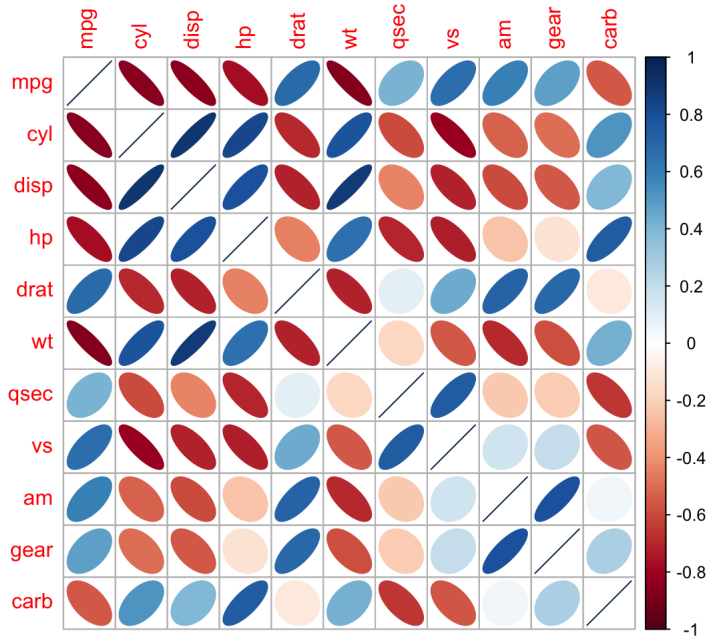


Don't you think this is a beautiful graph? If you look at the strip on the right, it is showing how each correlation value corresponds to a certain color on the red-blue spectrum. Red means a negative correlation and blue means a positive correlation. Now if you look at the square to the left, each circle indicates the correlation between a pair of variables. The larger the circle is, the stronger the correlation in absolute terms. On the diagonal, all the circels are of the biggest size with the darkest blue, indicating correlations of 1. This is because the correlation between a variable and itself mush be one. Now let's explore the options within corrplot.

Changing the Shapes

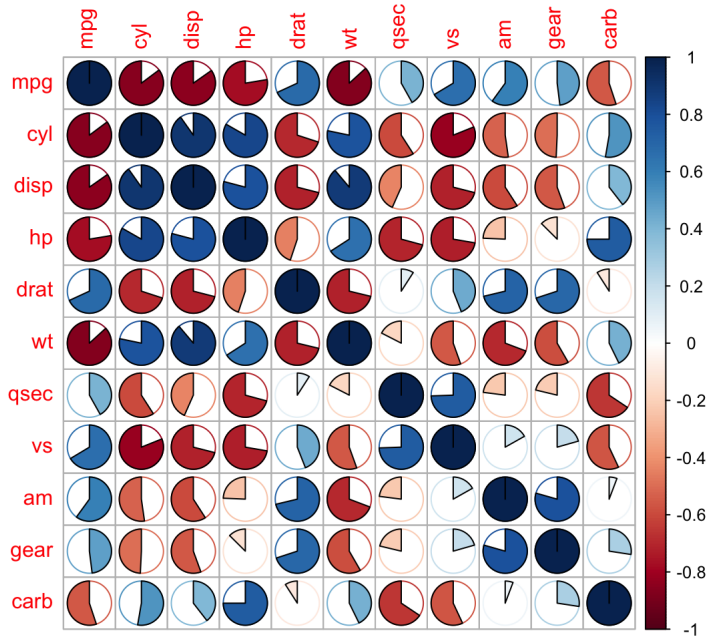
The "method" argument controls the shape used to display the correlations.

```
## method = "ellipse"
corrplot(plotted, method = "ellipse")
```



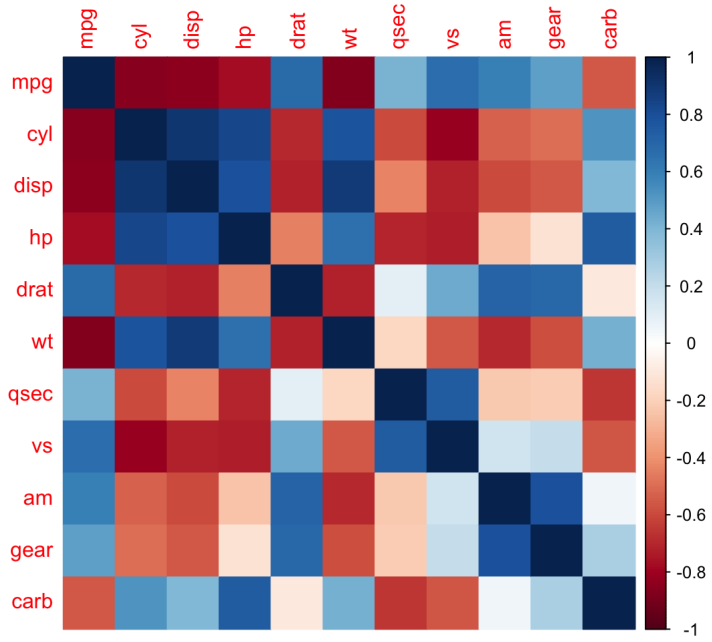
Similar to the previous graph, the circles are now replaced with ellipses. Rounder ellipses indicate smaller correlations in absolute terms. The orientation and color both indicate the sign of the correlation. Here are some other possible values of `method`:

```
# method = "pie"
corrplot(plotted, method = "pie")
```



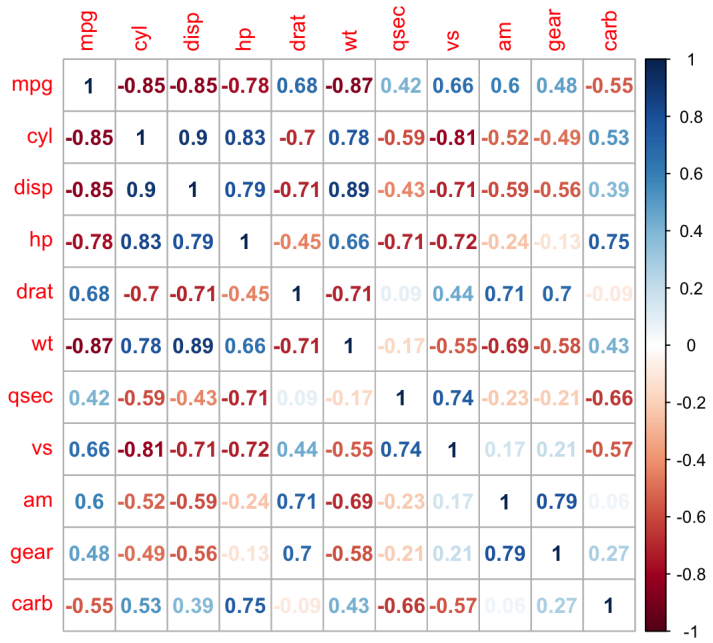
The color of the pie indicates the sign of the correlation, and the proportion of color within each pie indicates the magnitude. For instance, the number of cylinders in a car (cyl) has a strong positive correlation with the horsepower (hp).

```
# method = "color"
corrplot(plotted, method = "color")
```



Here, only the color of the squares indicate the magnitude and sign of the correlation.

```
## Method = "number"
corrplot(plotted, method = "number")
```



Of course, we can also choose to display the numbers directly. But this does not take full advantage of corrplot.

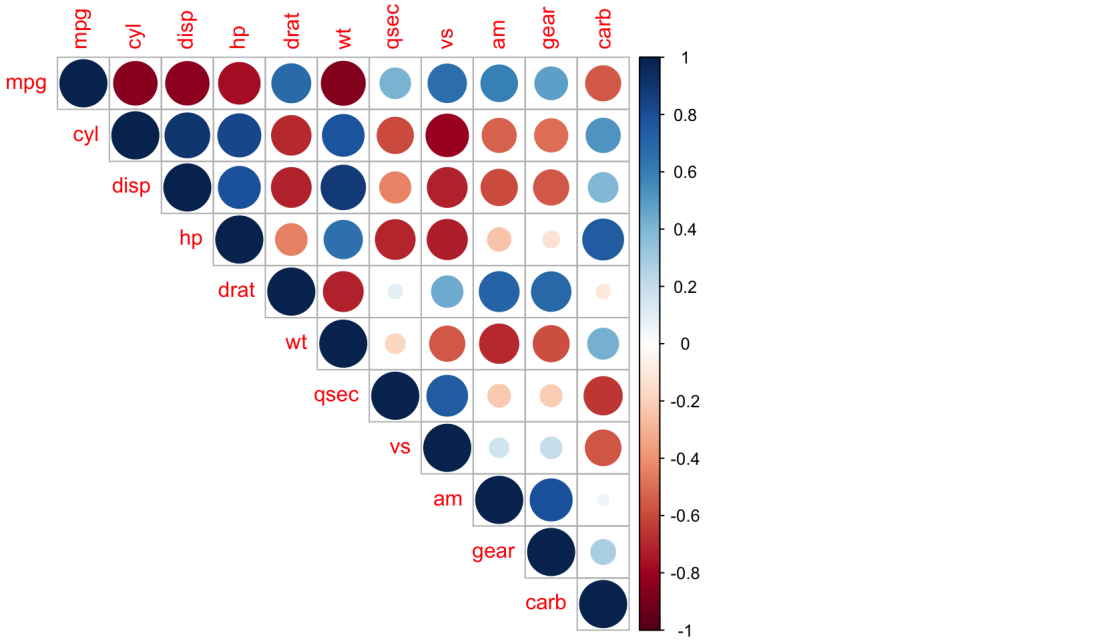
Correlogram Layouts

There are three general types of correlogram layouts:

- “full” (default, examples can be seen above) : display full correlation matrix
- “upper”: display upper triangular of the correlation matrix
- “lower”: display lower triangular of the correlation matrix.

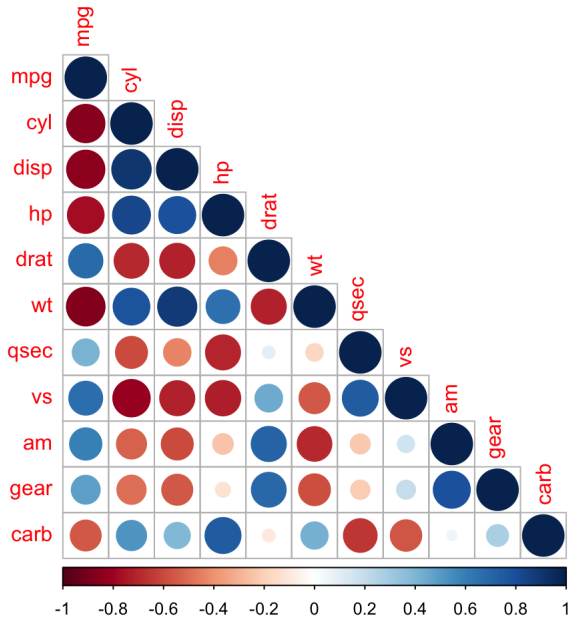
There will be an example for each of the types.

```
# upper triangular
corrplot(plotted, type = "upper")
```



Everything is the same as before, except for the fact that it is displyed in a triangular form. This exploits the symmetric nature of the correlation matrix. If I want to look at the correlation between miles per gallon and horsepower, it will be the fourth circle to the left of the first row.

```
#lower triangular
corrplot(plotted, type = "lower")
```

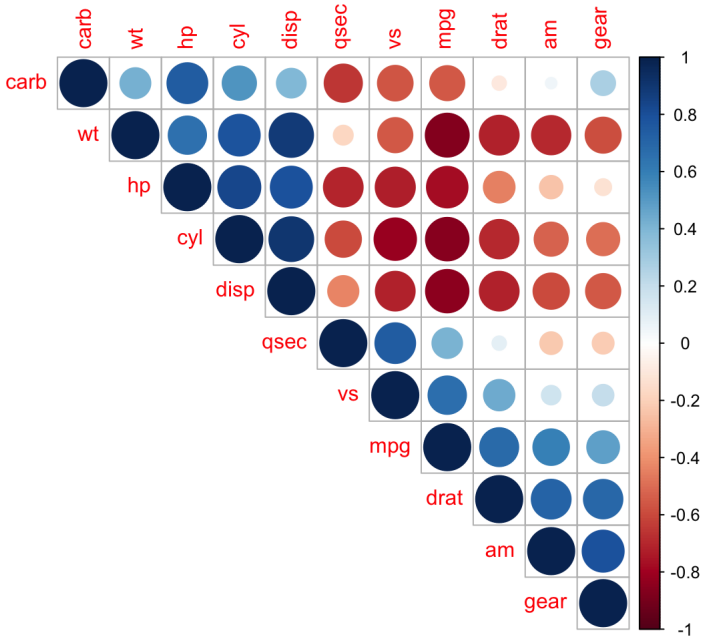


This is very similar to the graph above, besides the fact that points are concentrated in the lower half.

Reordering the Correlation Matrix

If you think the graphs above look messy and still take a lot of efforts to find the true patterns, you are not alone. This is why the argument "order" exists. The correlation matrix can be reordered according to the correlation coefficients with `order = "hclust"`. There are more advanced ways of ordering, but they are out of the scope of this post.

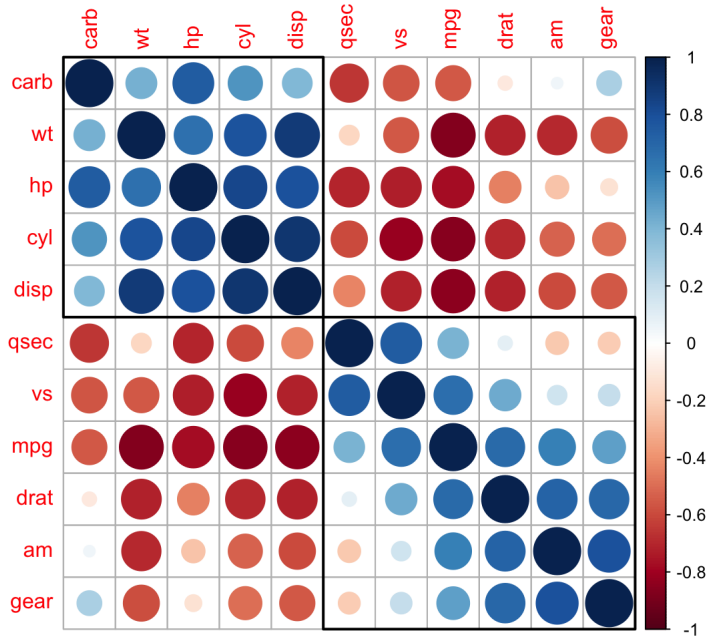
```
# correlogram with hclust reordering and upper-layout
corrplot(plotted, type = "upper", order = "hclust")
```



It is now very easy to tell which variables have the strongest correlations. We can now see easily that the number of cylinders (cyl) has a very negative correlation with mileage (mpg).

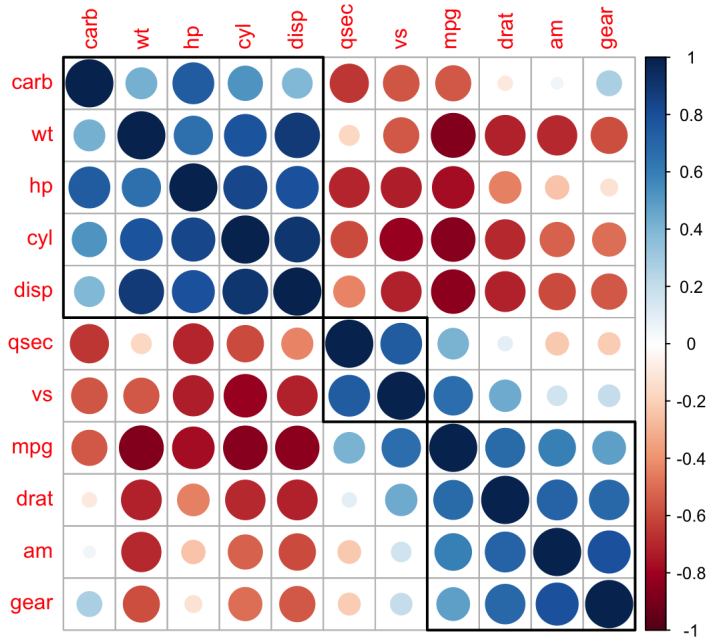
If we are using the order "hclust", we can actually draw rectangles around blocks based on hierarchical clustering.

```
## A correlogram ordered based on the correlation coefficients
corrplot(plotted, order = "hclust",

         ## This argument controls the number of rectangles
         addrect = 2)
```



As seen above, the areas where negative correlations are prevalent are highlighted by the black rectangles.

```
## A correlogram ordered based on the correlation coefficients
corrplot(plotted, order = "hclust",

         ## This argument controls the number of rectangles
         addrect = 3)
```

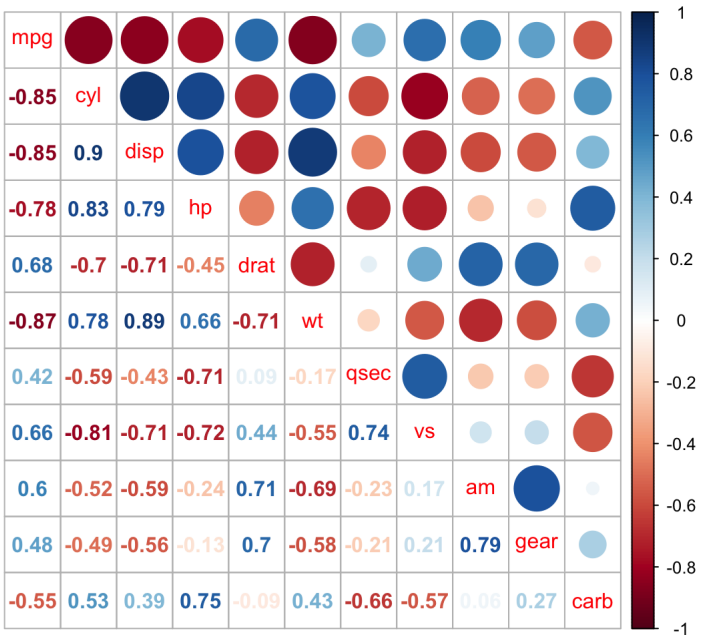


This graph puts three rectangles, dividing into more clusters. We can now better see that negative correlations are most prevalent around the diagonals of this matrix.

Mixed Layouts

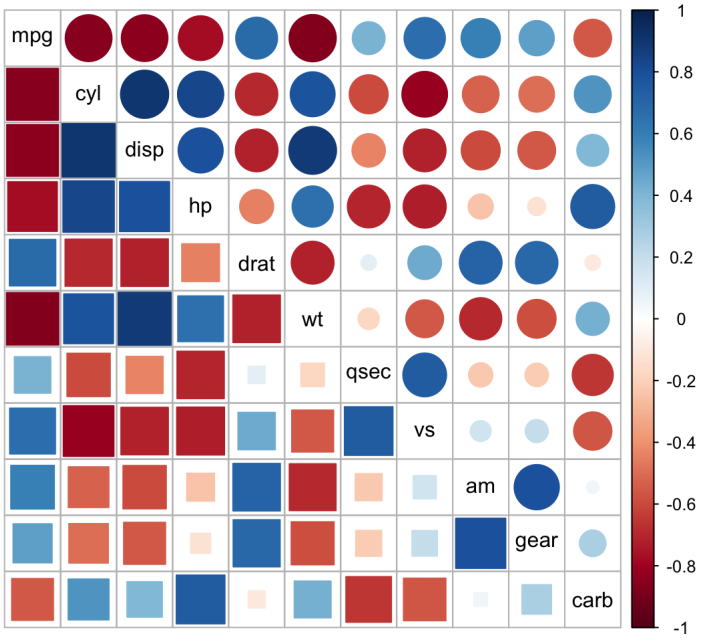
What if you like more than one layouts listed above? `corrplot.mixed()` is a wrapped function for mixed visualization style.

```
## Correlogram with numbers on the lower half and circles in the upper half
corrplot.mixed(plotted)
```



Clearly, this is a mix of numbers and circles. We can further choose what layouts to display.

```
## Correlogram with squares on the lower half and circles in the upper half
corrplot.mixed(plotted, lower = "square", upper = "circle",

               ## The argument tl.col controls the color of the label of the variables.
               tl.col = "black")
```

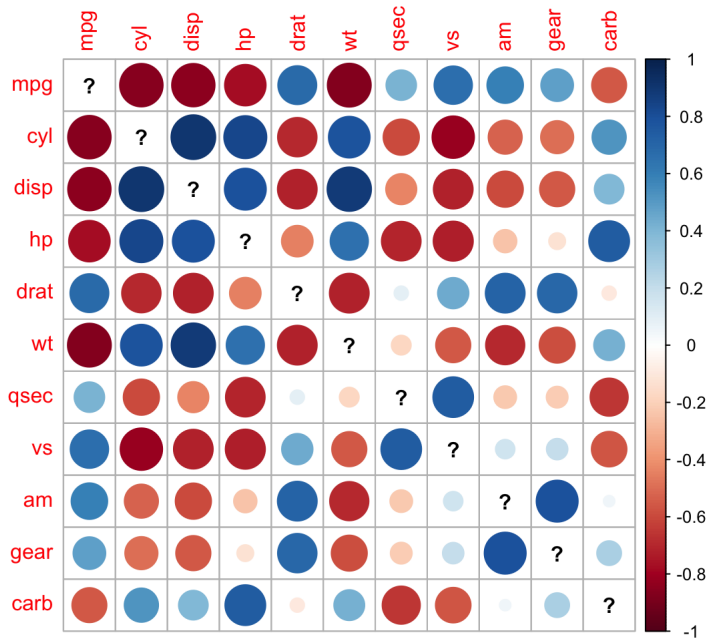


This is similar to the graph above, but shows the usage of more arguments.

Dealing with “NA” Values

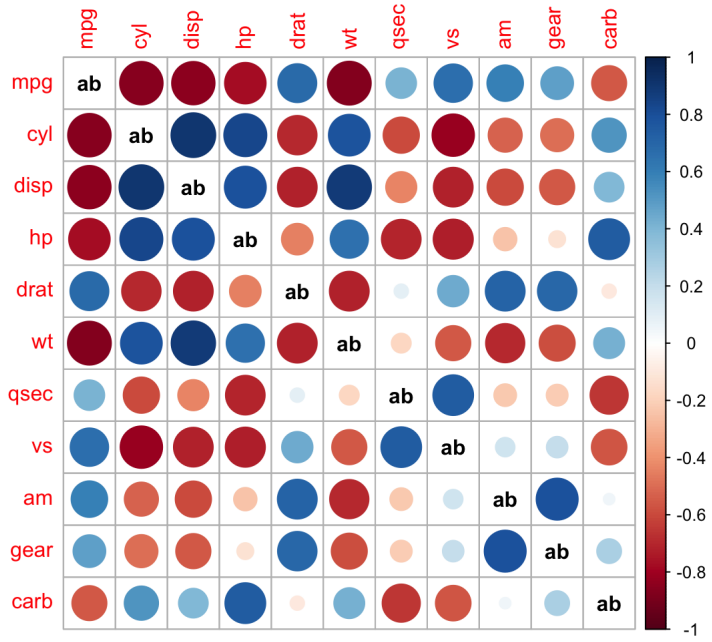
You might wonder, what if the data set we are given are not “clean” and there are missing values? Let’s begin by looking at how corrplot deals with it in default settings.

```
## Replacing certain values in the original correlation matrix with NA values.
plotted_NA = plotted
diag(plotted_NA) = NA

## Generating the correlogram
corrplot(plotted_NA)
```



As you can see, the missing values in the matrix is displayed as "?". This can be modified with the argument `na.label`:

```
## Generating the correlogram, with label "ab"
corrplot(plotted_NA, na.label = "ab")
```

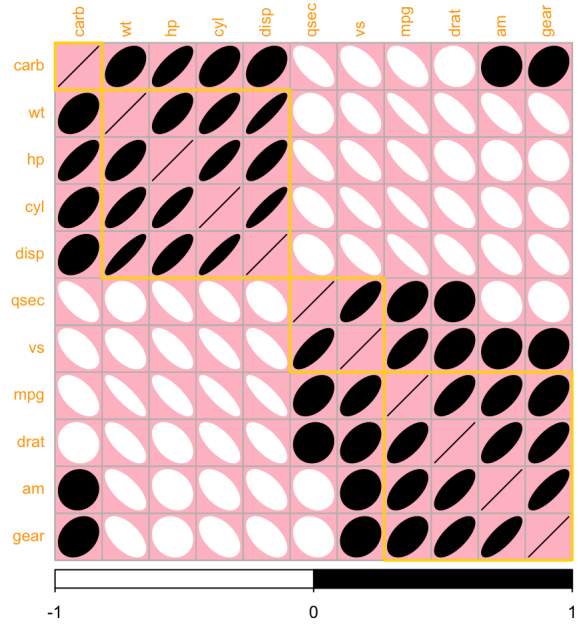


Now the question marks are replace with what we prefer, "ab". It is important to note that the length of the argument to na.label cannot exceed two characters.

Other Visual Elements

The example below will introduce some other arguments affecting the visualization of the correlation matrices and combine what we have learned so far.

```
corrplot(plotted,

         # Choosing the shape as ellipse
         method = "ellipse",

         # Ordering the matrix based on the correlation coefficients
         order = "hclust",

         # Adding four rectangular groupings based on the clustering
         addrect = 4,

         # Choosing the outline of the rectangle as gold
         rect.col = "gold",

         # Choosing the color of the shape to be black and white
         col = c("white", "black"),

         # Choosing the color in the background of the matrix as blue
         bg = "pink",

         # Choosing the color of the label of the variables as orange, and size as 0.75, which is smaller than the
default
         tl.col = "orange", tl.cex = 0.75,

         # Positioning the legend at the bottom
         cl.pos = "b"
         )
```



The graph reads similarly as the ones above, just with different visual elements.

Summary Message

I hope you gained a deeper understanding about corrplot through this post. Corrplot is indeed a powerful package, with many built-in visualization options, for visualizing the dependence between multiple variables at once.

References

1. https://cran.r-project.org/web/packages/corrplot/vignettes/corrplot-intro.html
2. https://cran.r-project.org/web/packages/corrplot/corrplot.pdf
3. http://rpubs.com/melike/corrplot
4. https://www.rdocumentation.org/packages/corrplot/versions/0.40/topics/corrplot
5. https://www.rdocumentation.org/packages/corrplot/versions/0.2-0/topics/corrplot
6. https://www.youtube.com/watch?v=jxUilFj2l-s
7. https://rstudio-pubs-static.s3.amazonaws.com/240657_5157ff98e8204c358b2118fa69162e18.html#what-is-a-correlation-matrix