Automating Data Exploration with R

Basic Exploration

Pair-wise correlations

Now that we have all our data transformed into numbers, we're ready to take a deeper look at it. Here we'll build two functions, a pair-wise correlation function and a more complicated function to get the p-value and correlation for each feature pair in the data.

But before we jump in, let's take a quick look at correlations. A great way to explore new data is to use a pairwise correlation matrix. This will measure the correlation between every combination of your variables. It doesn't really matter if you have an outcome (or response) variable at this point, it will compare everything against everything else.

For those not familiar with the correlation coefficient, it is simply a measure of similarity between two vectors of numbers. The measure value can range between 1 and -1, where 1 is perfectly correlated, -1 is perfectly inversly correlated, and 0 is not correlated at all:

```
print(cor(1:5,1:5))
```

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

```
print(cor(1:5,seq(100,500,100)))
```

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

```
print(cor(1:5,5:1))
```

```
## [1] -1
```

```
print(cor(1:5,c(1,2,3,4,4)))
```

```
## [1] 0.9701425
```

We'll use the `mtcars` data set that is already included in the **R** base package - it has the advantage of being fully numeric and clean.

```
# install.packages('dplyr')
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# install.packages('reshape2')
library(reshape2)

data_set <- mtcars
d_cor <- as.matrix(cor(data_set))
d_cor
```

```
##             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
```

For example, if you compare the two left utmost columns, we see that `mpg` is negatively correlated to `cycl`. Remember, correlations range from 1 to -1, with 1 being absolutely positively correlated, -1 being absolutely negatively correlated and 0 showing no correlations at all.

```
      mpg        cyl
```

mpg 1.0000000 -0.8521620

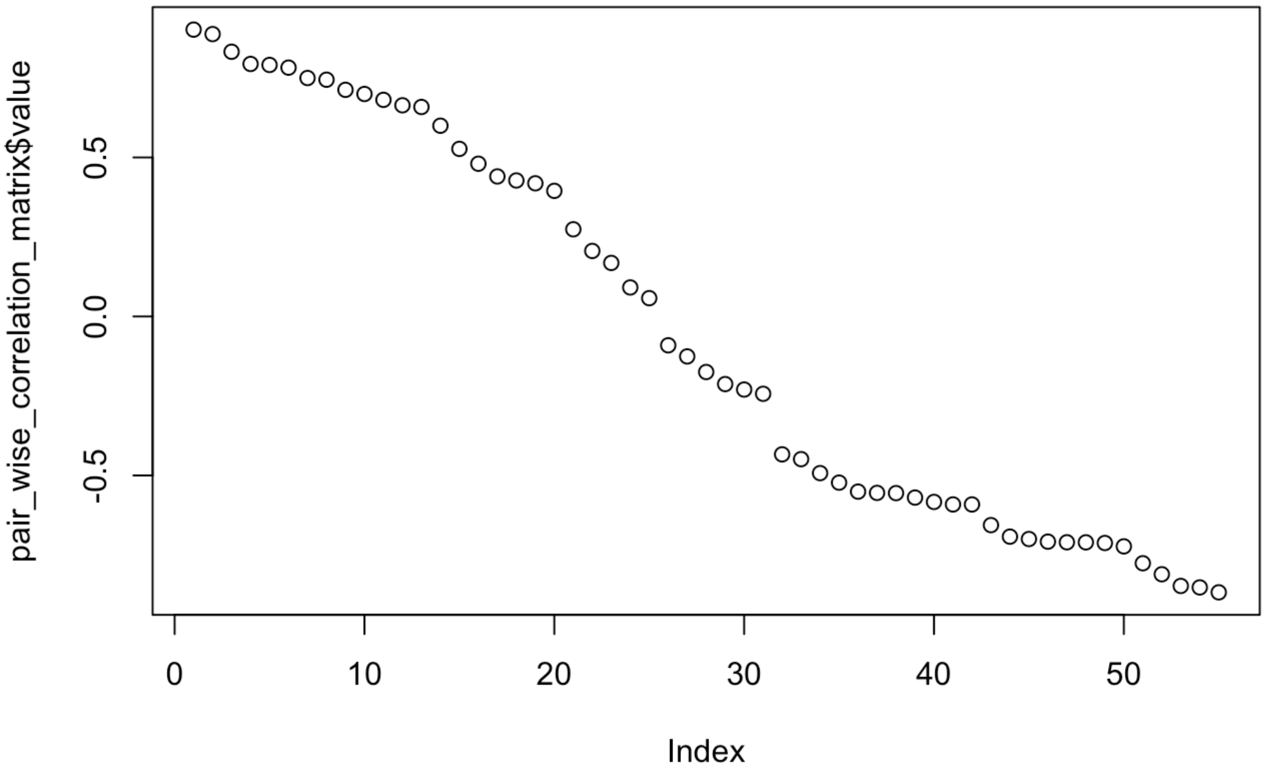
```r
d_cor_melt <- arrange(melt(d_cor), -(value))

# clean up
pair_wise_correlation_matrix <- filter(d_cor_melt, Var1 != Var2)
pair_wise_correlation_matrix <- filter(pair_wise_correlation_matrix, is.na(value)==FALSE)

# remove pair dups
dim(pair_wise_correlation_matrix)
```

```
## [1] 110   3
```

```r
pair_wise_correlation_matrix <- pair_wise_correlation_matrix[seq(1, nrow(pair_wise_correlation_matrix), by=2),]
dim(pair_wise_correlation_matrix)
```

```
## [1] 55  3
```

```r
plot(pair_wise_correlation_matrix$value)
```



Let’s build the above into a function:

```
Get_Fast_Correlations <- function(data_set, features_to_ignore=c(), size_cap=5000) {
     require(dplyr)
     require(reshape2)

     data_set <- data_set[,setdiff(names(data_set), features_to_ignore)]

     if (size_cap > nrow(data_set)) {
          data_set = data_set[sample(nrow(data_set), size_cap),]
     } else {
          data_set = data_set[sample(nrow(data_set), nrow(data_set)),]
     }

     d_cor <- as.matrix(cor(data_set))
     d_cor_melt <- arrange(melt(d_cor), -(value))

     # clean up
     pair_wise_correlation_matrix <- filter(d_cor_melt, Var1 != Var2)
     pair_wise_correlation_matrix <- filter(pair_wise_correlation_matrix, is.na(value)==FALSE)

     # remove pair dups
     dim(pair_wise_correlation_matrix)
     pair_wise_correlation_matrix <- pair_wise_correlation_matrix[seq(1, nrow(pair_wise_correlation_matrix), by=2),
]
     dim(pair_wise_correlation_matrix)

     plot(pair_wise_correlation_matrix$value)
     return(pair_wise_correlation_matrix)
}
```

We'll use the psych (https://cran.r-project.org/web/packages/psych/index.html) library to do a lot of the heavy lifting. We will delegate all the math to the `corr.test` function. It returns the following results (from the help files and plenty more there: ?corr.test):

- **r**: The matrix of correlations
- **n**: Number of cases per correlation
- **t**: Value of t-test for each correlation
- **p**: Two tailed probability of t for each correlation
- **se**: Standard error of the correlation
- **ci**: The alpha/2 lower and upper values

```r
# install.packages('psych')
library(psych)

data_set <- mtcars
featurenames_copy <- names(data_set)

# strip var names to index for pair wise identification
names(data_set) <- seq(1:ncol(data_set))
cor_data_df <-  corr.test(data_set)

# apply var names to correlation matrix over index
rownames(cor_data_df$r) <- featurenames_copy
colnames(cor_data_df$r) <- featurenames_copy

names(cor_data_df)
```

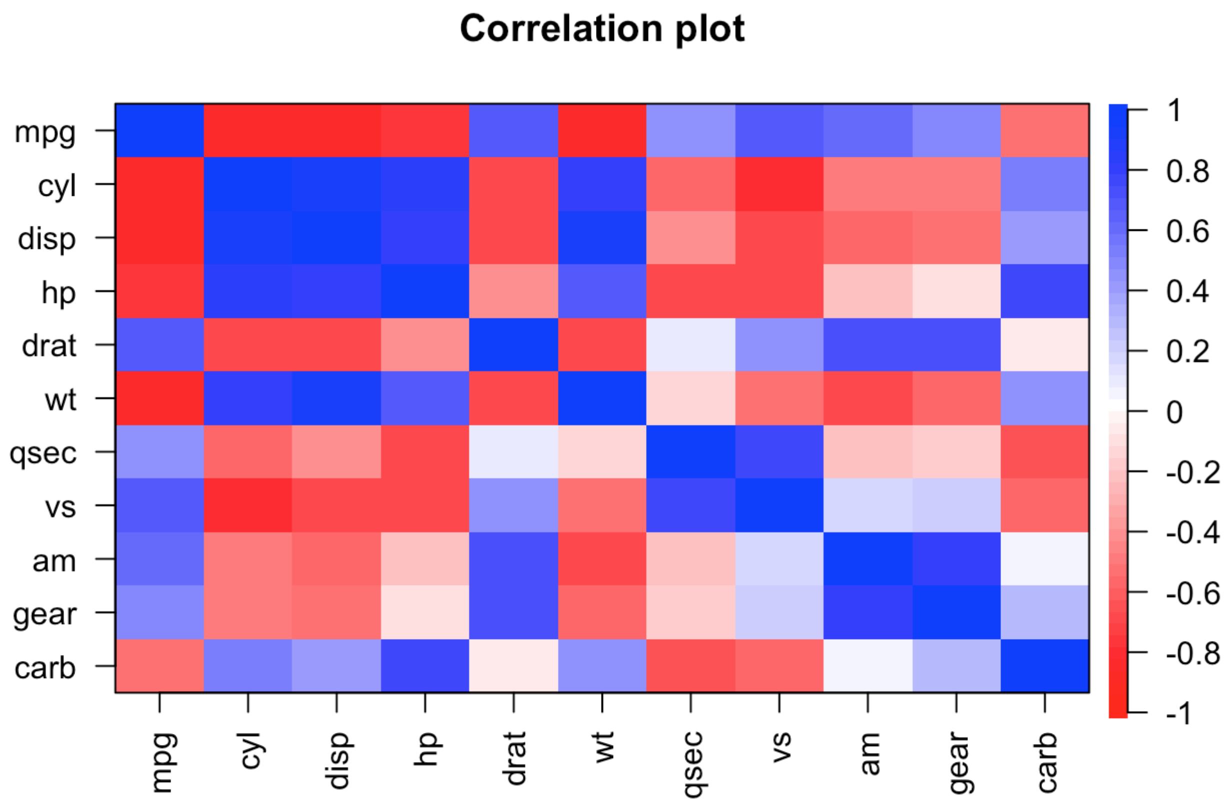
```
## [1] "r"      "n"      "t"      "p"      "se"     "adjust" "sym"    "ci"
## [9] "Call"
```

```r
# matrix of correlations
cor_data_df$r
```

```
##             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
```

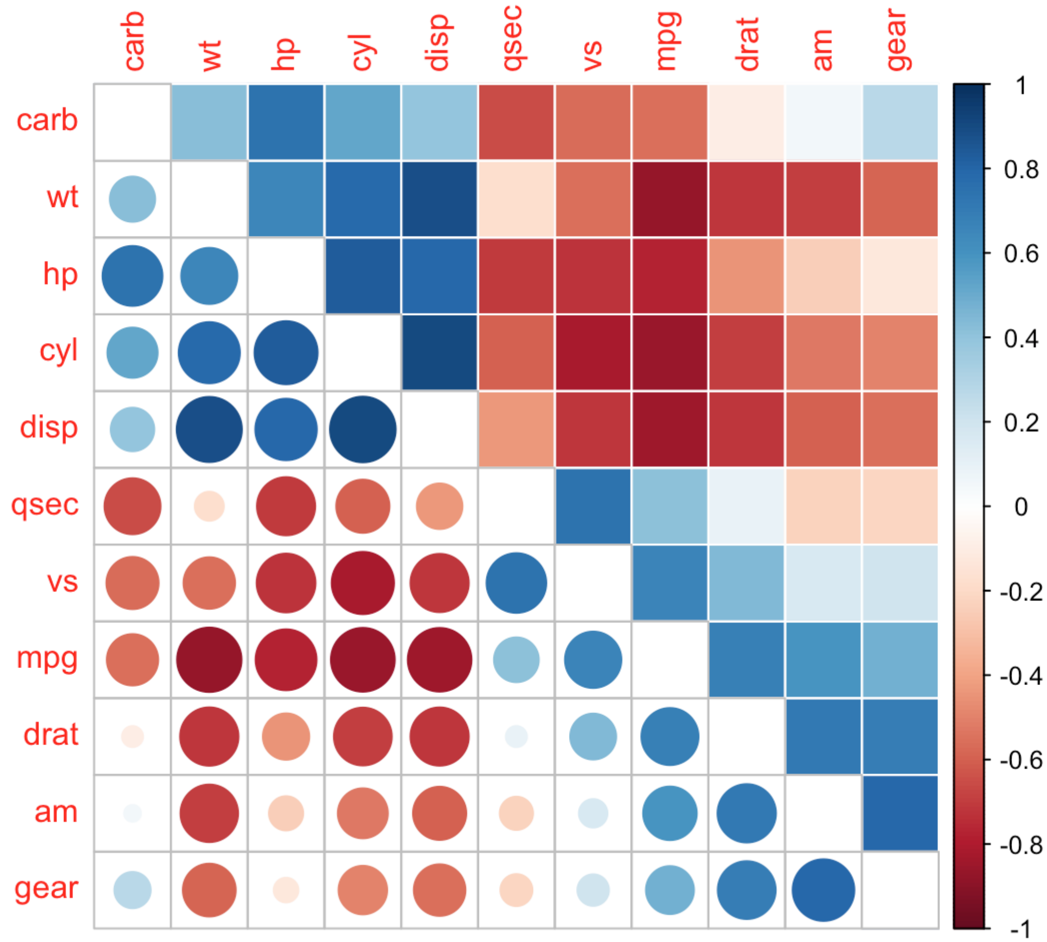
Let's visualize our correlation results using `cor.plot` from the psych (https://cran.r-project.org/web/packages/psych/index.html) library and `corrplot.mixed` from the corrplot (https://cran.r-project.org/web/packages/corrplot/vignettes/corrplot-intro.html)library.

```
cor.plot(cor_data_df$r)
```



Or even prettier with the `corrplot`` library:

```
#install.packages('corrplot')
library(corrplot)
corrplot.mixed(cor_data_df$r, lower="circle", upper="color",
tl.pos="lt", diag="n", order="hclust", hclust.method="complete")
```



We’re not going to add these plotting functions into our pipeline, but we will create a function using the `corr.test` results to highlight the strongest relationships.

Let’s build our pipeline function, discard all results except for the significance (p-value) and correlations, and set our cut-off thresholds:

```r
Get_Top_Relationships <- function(data_set, correlation_abs_threshold=0.8,
                                  pvalue_threshold=0.01) {
     require(psych)
     require(dplyr)

     feature_names <- names(data_set)

     # strip var names to index for pair-wise identification
     names(data_set) <- seq(1:ncol(data_set))
     # calculate correlation and significance numbers
     cor_data_df <-  corr.test(data_set)

     # apply var names to correlation matrix over index
     rownames(cor_data_df$r) <- feature_names
     colnames(cor_data_df$r) <- feature_names

     # top cor and sig
     relationships_set <- cor_data_df$ci[,c('r','p')]

     # apply var names to data over index pairs
     relationships_set$feature_1 <- feature_names[as.numeric(sapply(strsplit(rownames(relationships_set), "-"), `
[`, 1))]
     relationships_set$feature_2 <- feature_names[as.numeric(sapply(strsplit(rownames(relationships_set), "-"), `
[`, 2))]

     relationships_set <- select(relationships_set, feature_1, feature_2, r, p) %>%
          rename(correlaton=r, pvalue=p)

     # return only the most insteresting relationships
     return(filter(relationships_set, abs(correlaton) > correlation_abs_threshold |
                        pvalue < pvalue_threshold) %>% arrange(pvalue))

}

dim(Get_Top_Relationships(mtcars))
```

```
## [1] 39  4
```

```
head(Get_Top_Relationships(mtcars))
```

```
##   feature_1 feature_2 correlaton       pvalue
## 1       cyl      disp  0.9020329 1.803002e-12
## 2      disp        wt  0.8879799 1.222311e-11
## 3       mpg        wt -0.8676594 1.293958e-10
## 4       mpg       cyl -0.8521620 6.112688e-10
## 5       mpg      disp -0.8475514 9.380328e-10
## 6       cyl        hp  0.8324475 3.477861e-09
```