2 Hands On: Data Quality and Pre-Processing

1. Assessing Data Quality

Load the carInsurance data set about the insurance risk rating of cars based on several characteristics of each car.

(a) Check if there are any missing values.

```
load("C:/Proyectos ML/Mineria/Hands_On_2/data/carInsurance.Rdata")
data <- as.data.frame(carIns)
has_missing_values <- anyNA(data)
print(has_missing_values)
```

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

(b) Count the number of cases that have, at least, one missing value.

```
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
```

```
data <- as.data.frame(carIns)
filas<- nrow(data)
print(sprintf("Número de filas totales %d",filas))
```

```
## [1] "Número de filas totales 205"
```

```
filtered_data <- data %>% stats::filter(complete.cases(.))
filas_completas <-nrow(filtered_data)
print(sprintf("Número de filas completas %d",filas_completas))
```

```
## [1] "Número de filas completas 205"
```

(c) Create a new data set by removing all the cases that have missing values.

```r
library(dplyr)

df <- as.data.frame(carIns)
nuevaData <- df[complete.cases(df) ==  TRUE,]
#print(filtrar)
library(flextable)
```

```
## Warning: package 'flextable' was built under R version 4.3.1
```

```r
datos <- flextable(nuevaData)
new_tabla<- autofit(datos)
print(new_tabla)
```

```
## a flextable object.
## col_keys: `symb`, `normLoss`, `make`, `fuelType`, `aspiration`, `nDoors`, `bodyStyle`, `driveWheels`
## header has 1 row(s)
## body has 159 row(s)
## original dataset sample:
##    symb normLoss make fuelType aspiration nDoors bodyStyle driveWheels
## 4     2      164 audi      gas        std   four     sedan         fwd
## 5     2      164 audi      gas        std   four     sedan         4wd
## 7     1      158 audi      gas        std   four     sedan         fwd
## 9     1      158 audi      gas      turbo   four     sedan         fwd
## 11    2      192  bmw      gas        std    two     sedan         rwd
##    engineLocation wheelBase length width height curbWeight engineType nrCylinds
## 4           front      99.8  176.6  66.2   54.3       2337        ohc      four
## 5           front      99.4  176.6  66.4   54.3       2824        ohc      five
## 7           front     105.8  192.7  71.4   55.7       2844        ohc      five
## 9           front     105.8  192.7  71.4   55.9       3086        ohc      five
## 11          front     101.2  176.8  64.8   54.3       2395        ohc      four
##    engineSize fuelSystem bore stroke compressionRatio horsePower peakRpm
## 4         109       mpfi 3.19    3.4             10.0        102    5500
## 5         136       mpfi 3.19    3.4              8.0        115    5500
## 7         136       mpfi 3.19    3.4              8.5        110    5500
## 9         131       mpfi 3.13    3.4              8.3        140    5500
## 11        108       mpfi 3.50    2.8              8.8        101    5800
##    cityMpg highwayMpg price
## 4       24         30 13950
## 5       18         22 17450
## 7       19         25 17710
## 9       17         20 23875
## 11      23         29 16430
```

(d) Create a new data set by imputing all the missing values with 0.

```r
#install.packages("tidyimpute")
library(tidyimpute)
datos_sin_cero<- carIns %>%na.omit()
# Cree un nuevo conjunto de datos con valores faltantes imputados como 0
imputed_data <- impute(datos_sin_cero, method = "fixed", fixed_value = 0)

# View the imputed dataset
head(imputed_data)
```

```
## # A tibble: 6 x 26
##    symb normLoss make  fuelType aspiration nDoors bodyStyle driveWheels
##   <int>    <int> <fct> <fct>    <fct>      <fct>  <fct>     <fct>
## 1     2      164 audi  gas      std        four   sedan     fwd
## 2     2      164 audi  gas      std        four   sedan     4wd
## 3     1      158 audi  gas      std        four   sedan     fwd
## 4     1      158 audi  gas      turbo      four   sedan     fwd
## 5     2      192 bmw   gas      std        two    sedan     rwd
## 6     0      192 bmw   gas      std        four   sedan     rwd
## # i 18 more variables: engineLocation <fct>, wheelBase <dbl>, length <dbl>,
## #   width <dbl>, height <dbl>, curbWeight <int>, engineType <fct>,
## #   nrCylinds <fct>, engineSize <int>, fuelSystem <fct>, bore <dbl>,
## #   stroke <dbl>, compressionRatio <dbl>, horsePower <int>, peakRpm <int>,
## #   cityMpg <int>, highwayMpg <int>, price <int>
```

(e) Create a new data set by imputing the mean in all the columns which have double type values

```
library(dplyr)
datos<- carIns%>%filter(complete.cases(.))
# Create a new dataset by imputing the mean in columns with double type values
imputed_data <-  datos %>%mutate_if(is.double, ~ ifelse(is.na(.), mean(., na.rm = TRUE), .))
head(imputed_data)
```

```
## # A tibble: 6 x 26
##    symb normLoss make  fuelType aspiration nDoors bodyStyle driveWheels
##   <int>    <int> <fct> <fct>    <fct>      <fct>  <fct>     <fct>
## 1     2      164 audi  gas      std        four   sedan     fwd
## 2     2      164 audi  gas      std        four   sedan     4wd
## 3     1      158 audi  gas      std        four   sedan     fwd
## 4     1      158 audi  gas      turbo      four   sedan     fwd
## 5     2      192 bmw   gas      std        two    sedan     rwd
## 6     0      192 bmw   gas      std        four   sedan     rwd
## # i 18 more variables: engineLocation <fct>, wheelBase <dbl>, length <dbl>,
## #   width <dbl>, height <dbl>, curbWeight <int>, engineType <fct>,
## #   nrCylinds <fct>, engineSize <int>, fuelSystem <fct>, bore <dbl>,
## #   stroke <dbl>, compressionRatio <dbl>, horsePower <int>, peakRpm <int>,
## #   cityMpg <int>, highwayMpg <int>, price <int>
```

(f) Create a new data set by imputing the mode in all the columns which have integer type values.

```
# Identify names columns with integer type values
eliminar_na <- carIns %>%filter(complete.cases(.))
integer_columns <- sapply(eliminar_na, is.integer)
seleccionar_columnas <- names(integer_columns[integer_columns])
print(seleccionar_columnas)
```

```
## [1] "symb"       "normLoss"   "curbWeight" "engineSize" "horsePower"
## [6] "peakRpm"    "cityMpg"    "highwayMpg" "price"
```

```
# Funcion para calcular la moda de un vector
Mode <- function(x) {
  ux <- unique(x)
```

```
  ux[which.max(tabulate(match(x, ux)))]
}
# Calcular la moda para cada entero en la columna
modes <- sapply(eliminar_na[, integer_columns], Mode)

# Imprimir el nombre de la columna con su respectiva moda
for (i in 1:length(modes)) {
  print(paste("Column:", names(modes)[i], "Mode:", modes[i]))
}
```

```
## [1] "Column: symb Mode: 0"
## [1] "Column: normLoss Mode: 161"
## [1] "Column: curbWeight Mode: 1989"
## [1] "Column: engineSize Mode: 92"
## [1] "Column: horsePower Mode: 68"
## [1] "Column: peakRpm Mode: 4800"
## [1] "Column: cityMpg Mode: 31"
## [1] "Column: highwayMpg Mode: 38"
## [1] "Column: price Mode: 5572"
```

(g) Create a new data set by imputing the most frequent value to the column nDoors

```
# Load the required packages
library(dplyr)

# Load the carInsurance dataset
eliminar_na <- carIns%>%na.omit()

# Define the Mode function to calculate the mode of a vector
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

# Create a new data set by imputing the most frequent value to the column "nDoors"
new_carInsurance <- carIns %>%
  mutate(nDoors = ifelse(is.na(nDoors), Mode(nDoors), nDoors))

# Print the new data set
print(new_carInsurance)
```

```
## # A tibble: 205 x 26
##     symb normLoss make        fuelType aspiration nDoors bodyStyle   driveWheels
##    <int>    <int> <fct>       <fct>    <fct>       <int> <fct>       <fct>
##  1     3       NA alfa-romero gas      std             2 convertible rwd
##  2     3       NA alfa-romero gas      std             2 convertible rwd
##  3     1       NA alfa-romero gas      std             2 hatchback   rwd
##  4     2      164 audi        gas      std             1 sedan       fwd
##  5     2      164 audi        gas      std             1 sedan       4wd
##  6     2       NA audi        gas      std             2 sedan       fwd
##  7     1      158 audi        gas      std             1 sedan       fwd
##  8     1       NA audi        gas      std             1 wagon       fwd
```

```
##  9     1      158 audi        gas      turbo           1 sedan       fwd
## 10     0       NA audi        gas      turbo           2 hatchback   4wd
## # i 195 more rows
## # i 18 more variables: engineLocation <fct>, wheelBase <dbl>, length <dbl>,
## #   width <dbl>, height <dbl>, curbWeight <int>, engineType <fct>,
## #   nrCylinds <fct>, engineSize <int>, fuelSystem <fct>, bore <dbl>,
## #   stroke <dbl>, compressionRatio <dbl>, horsePower <int>, peakRpm <int>,
## #   cityMpg <int>, highwayMpg <int>, price <int>
```

(h) Combine the three last imputations to obtain a final dataset. Are there any duplicated cases?

```r
# Load the required packages
library(dplyr)

# Define the Mode function to calculate the mode of a vector
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

# Impute missing values in columns with integer type using mode
carInsurance_imputed_mode <- carIns %>%
  mutate(across(where(is.integer), ~ ifelse(is.na(.), Mode(.), .)))

# Impute missing values in the "nDoors" column using the most frequent value
carInsurance_imputed_nDoors <- carInsurance_imputed_mode %>%
  mutate(nDoors = ifelse(is.na(nDoors), Mode(nDoors), nDoors))

# Convert the "nDoors" column to character type in both datasets
carInsurance_imputed_mode$nDoors <- as.character(carInsurance_imputed_mode$nDoors)
carInsurance_imputed_nDoors$nDoors <- as.character(carInsurance_imputed_nDoors$nDoors)

# Combine the imputed datasets to obtain a final dataset
final_carInsurance <- bind_rows(carInsurance_imputed_mode, carInsurance_imputed_nDoors)

# Check for duplicated cases
duplicated_cases <- final_carInsurance %>%
  distinct() %>%
  count() %>%
  filter(n > 1)

# Print the duplicated cases
print(duplicated_cases)
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1   410
```

2. Data Pre-Processing

Leer the package dlookr. Use the same car insurance data set above and apply the following transformations to the price attribute. Be critical regarding the obtained results.

(a)Apply range-based normalization and z-score normalization.

```
library(dplyr)

normalizar<-carIns %>%  mutate(
          # Aplica la normalizaciÃ³n basada en rangos
          precioNormRango = (price - min(price, na.rm = TRUE)) / (max(price, na.rm = TRUE) - min(price,
          # Aplica la normalizaciÃ³n Z-score
          precioNormZscore = (price - mean(price, na.rm = TRUE)) / sd(price, na.rm = TRUE)
)
print(normalizar)
```

```
## # A tibble: 205 x 28
##     symb normLoss make        fuelType aspiration nDoors bodyStyle   driveWheels
##    <int>    <int> <fct>       <fct>    <fct>      <fct>  <fct>       <fct>
##  1     3       NA alfa-romero gas      std        two    convertible rwd
##  2     3       NA alfa-romero gas      std        two    convertible rwd
##  3     1       NA alfa-romero gas      std        two    hatchback   rwd
##  4     2      164 audi        gas      std        four   sedan       fwd
##  5     2      164 audi        gas      std        four   sedan       4wd
##  6     2       NA audi        gas      std        two    sedan       fwd
##  7     1      158 audi        gas      std        four   sedan       fwd
##  8     1       NA audi        gas      std        four   wagon       fwd
##  9     1      158 audi        gas      turbo      four   sedan       fwd
## 10     0       NA audi        gas      turbo      two    hatchback   4wd
## # i 195 more rows
## # i 20 more variables: engineLocation <fct>, wheelBase <dbl>, length <dbl>,
## #   width <dbl>, height <dbl>, curbWeight <int>, engineType <fct>,
## #   nrCylinds <fct>, engineSize <int>, fuelSystem <fct>, bore <dbl>,
## #   stroke <dbl>, compressionRatio <dbl>, horsePower <int>, peakRpm <int>,
## #   cityMpg <int>, highwayMpg <int>, price <int>, precioNormRango <dbl>,
## #   precioNormZscore <dbl>
```

(b) Discretize it into 4 equal-frequency ranges an into 4 equal-width ranges.

```
carIns2<-carIns %>%na.omit()

# Discretize price into 4 equal-frequency ranges
carIns2$price_freq <- cut(carIns2$price, breaks = quantile(carIns2$price, probs = seq(0, 1, by = 0.25),

# Discretize price into 4 equal-width ranges
carIns2$price_width <- cut(carIns2$price, breaks = 4, labels = FALSE, include.lowest = TRUE)

# Print the transformed data set
print(carIns2)
```

```
## # A tibble: 159 x 28
##     symb normLoss make      fuelType aspiration nDoors bodyStyle driveWheels
##    <int>    <int> <fct>     <fct>    <fct>      <fct>  <fct>     <fct>
##  1     2      164 audi      gas      std        four   sedan     fwd
##  2     2      164 audi      gas      std        four   sedan     4wd
##  3     1      158 audi      gas      std        four   sedan     fwd
##  4     1      158 audi      gas      turbo      four   sedan     fwd
##  5     2      192 bmw       gas      std        two    sedan     rwd
```

```
##  6     0      192 bmw       gas      std        four   sedan     rwd
##  7     0      188 bmw       gas      std        two    sedan     rwd
##  8     0      188 bmw       gas      std        four   sedan     rwd
##  9     2      121 chevrolet gas      std        two    hatchback fwd
## 10     1       98 chevrolet gas      std        two    hatchback fwd
## # i 149 more rows
## # i 20 more variables: engineLocation <fct>, wheelBase <dbl>, length <dbl>,
## #   width <dbl>, height <dbl>, curbWeight <int>, engineType <fct>,
## #   nrCylinds <fct>, engineSize <int>, fuelSystem <fct>, bore <dbl>,
## #   stroke <dbl>, compressionRatio <dbl>, horsePower <int>, peakRpm <int>,
## #   cityMpg <int>, highwayMpg <int>, price <int>, price_freq <int>,
## #   price_width <int>
```

3. With the seed 111019 obtain the following samples on the car insurance data set.

(a) A random sample of 60% of the cases, with replacement

```
library(dplyr)
muestra_60 <- sample_frac(carIns, 0.6)
print(muestra_60)
```

```
## # A tibble: 123 x 26
##     symb normLoss make        fuelType aspiration nDoors bodyStyle   driveWheels
##    <int>    <int> <fct>       <fct>    <fct>      <fct>  <fct>       <fct>
##  1     0       91 toyota      gas      std        four   sedan       fwd
##  2     1      231 nissan      gas      std        two    hatchback   rwd
##  3     3       NA alfa-romero gas      std        two    convertible rwd
##  4     2       NA isuzu       gas      std        two    hatchback   rwd
##  5     0      102 subaru      gas      std        four   sedan       4wd
##  6     0       NA volkswagen  diesel   turbo      four   sedan       fwd
##  7     1       NA isuzu       gas      std        two    sedan       fwd
##  8     2      137 honda       gas      std        two    hatchback   fwd
##  9     0      106 honda       gas      std        two    hatchback   fwd
## 10     3      197 toyota      gas      std        two    hatchback   rwd
## # i 113 more rows
## # i 18 more variables: engineLocation <fct>, wheelBase <dbl>, length <dbl>,
## #   width <dbl>, height <dbl>, curbWeight <int>, engineType <fct>,
## #   nrCylinds <fct>, engineSize <int>, fuelSystem <fct>, bore <dbl>,
## #   stroke <dbl>, compressionRatio <dbl>, horsePower <int>, peakRpm <int>,
## #   cityMpg <int>, highwayMpg <int>, price <int>
```

(b) A stratified sample of 60% of the cases of cars, according to the fuelType attribute

```
# Install and load the splitstackshape package
#install.packages("splitstackshape")
library(splitstackshape)
set.seed(111019)
strat_campo <- stratified(carIns, "fuelType", 0.6)

print(strat_campo)
```

```
##      symb normLoss          make fuelType aspiration nDoors bodyStyle
```

```
##   1:    1      103        nissan      gas        std   four     wagon
##   2:    1      101         honda      gas        std    two hatchback
##   3:    3       NA       porsche      gas        std    two   hardtop
##   4:    2       NA          audi      gas        std    two     sedan
##   5:    0       91        toyota      gas        std   four     sedan
##  ---
## 119:    0      161        peugot   diesel      turbo   four     sedan
## 120:   -1       93 mercedes-benz   diesel      turbo   four     wagon
## 121:   -1       65        toyota   diesel      turbo   four     sedan
## 122:    0       NA        peugot   diesel      turbo   four     wagon
## 123:    0       NA    volkswagen   diesel      turbo   four     sedan
##      driveWheels engineLocation wheelBase length width height curbWeight
##   1:         fwd          front      94.5  170.2  63.8   53.5       2037
##   2:         fwd          front      93.7  150.0  64.0   52.6       1940
##   3:         rwd           rear      89.5  168.9  65.0   51.6       2756
##   4:         fwd          front      99.8  177.3  66.3   53.1       2507
##   5:         fwd          front      95.7  166.3  64.4   53.0       2081
##  ---
## 119:         rwd          front     107.9  186.7  68.4   56.7       3197
## 120:         rwd          front     110.0  190.9  70.3   58.7       3750
## 121:         fwd          front     102.4  175.6  66.5   54.9       2480
## 122:         rwd          front     114.2  198.9  68.4   58.7       3430
## 123:         fwd          front     100.4  180.2  66.9   55.1       2579
##      engineType nrCylinds engineSize fuelSystem bore stroke compressionRatio
##   1:        ohc      four         97       2bbl 3.15   3.29              9.4
##   2:        ohc      four         92       1bbl 2.91   3.41              9.2
##   3:       ohcf       six        194       mpfi 3.74   2.90              9.5
##   4:        ohc      five        136       mpfi 3.19   3.40              8.5
##   5:        ohc      four         98       2bbl 3.19   3.03              9.0
##  ---
## 119:          l      four        152        idi 3.70   3.52             21.0
## 120:        ohc      five        183        idi 3.58   3.64             21.5
## 121:        ohc      four        110        idi 3.27   3.35             22.5
## 122:          l      four        152        idi 3.70   3.52             21.0
## 123:        ohc      four         97        idi 3.01   3.40             23.0
##      horsePower peakRpm cityMpg highwayMpg price
##   1:         69    5200      31         37  7999
##   2:         76    6000      30         34  6529
##   3:        207    5900      17         25 34028
##   4:        110    5500      19         25 15250
##   5:         70    4800      30         37  6938
##  ---
## 119:         95    4150      28         33 13200
## 120:        123    4350      22         25 28248
## 121:         73    4500      30         33 10698
## 122:         95    4150      25         25 13860
## 123:         68    4500      33         38 13845
```

(c) Use the table() function to inspect the distribution of values in each of the two samples above.

```
load("C:/Proyectos ML/Mineria/Hands_On_2/data/carIns_final.Rdata")
data <- as.data.frame(carIns_final)
# Set the seed
# Set the seed
```

```r
set.seed(111019)

# Obtain a random sample of 60% of the cases with replacement
muestra_data_1 <- sample_frac(data, 0.6, replace = TRUE)$fuelType

# Obtain a random sample of 40% of the cases with replacement
muestra_data_2 <- sample_frac(data, 0.4, replace = TRUE)$make

# Inspect the distribution of values in each sample
table_sample_1 <- table(muestra_data_1)
table_sample_2 <- table(muestra_data_2)

# View the distributions
cat("Combustible\n")
```

```
## Combustible
```

```r
print(table_sample_1)
```

```
## muestra_data_1
## diesel    gas
##      8    115
```

```r
cat("\nTipo auto\n")
```

```
##
## Tipo auto
```

```r
print(table_sample_2)
```

```
## muestra_data_2
##   alfa-romero          audi           bmw     chevrolet         dodge
##             1             4             1             0             4
##         honda         isuzu        jaguar         mazda mercedes-benz
##             2             4             1             3             2
##       mercury    mitsubishi        nissan        peugot      plymouth
##             1             4             8             8             8
##       porsche       renault          saab        subaru        toyota
##             3             0             2             4             7
##    volkswagen         volvo
##             6             9
```

4. Load the package corrplot and select the numeric attributes of the car insurance data set.

(a) Using the function cor(), obtain the pearson correlation coefficient between each pair of variables.

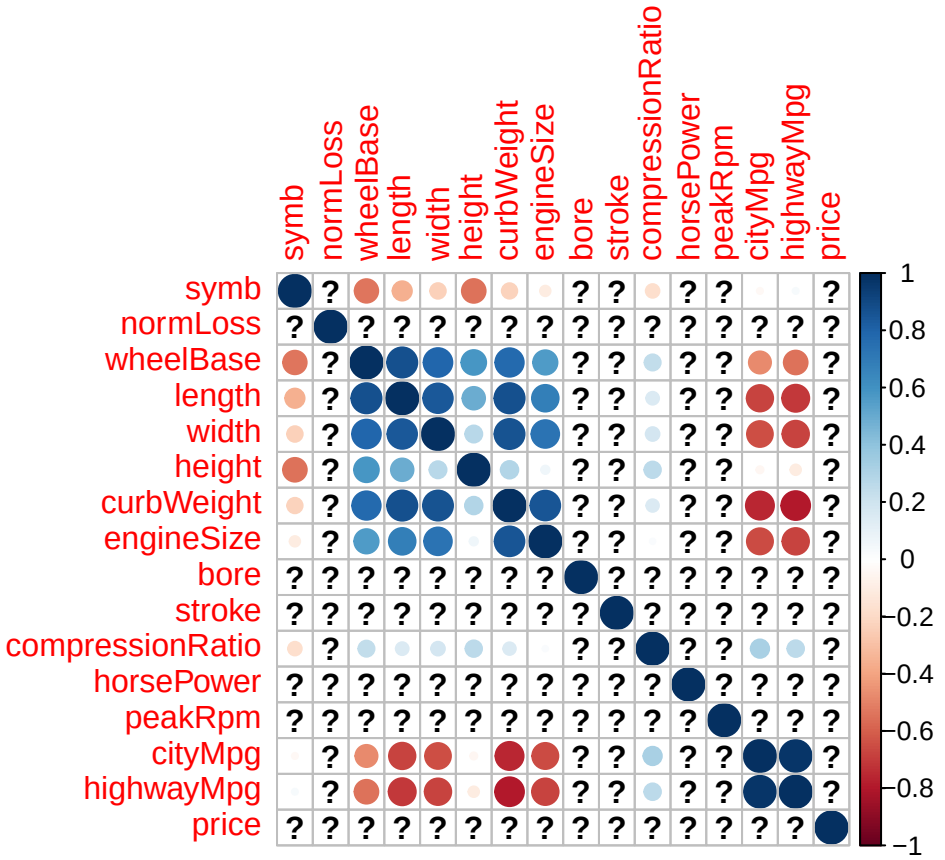
```r
#install.packages("corrplot")
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.3.1
```

```
## corrplot 0.92 loaded
```

```
numeric_data <- carIns[, sapply(carIns, is.numeric)]
cor_matrix <- cor(numeric_data)

corrplot(cor_matrix, method = "circle")
```



(b) Apply the function cor.mtest() to the previous result to calculate the p-values and confidence intervals of the correlation coefficient for each pair of variables.

```
library(corrplot)
numeric_data <- carIns[, sapply(carIns, is.numeric)]

cor_matrix <- cor(numeric_data)

library(psych)
```

```
## Warning: package 'psych' was built under R version 4.3.1
```

```
cor_test <- cor.mtest(numeric_data)


p_values <- cor_test$p

conf_intervals <- cor_test$uppCI
```

```
cat("\n p_values ")
```

```
##
##  p_values
```

```
cat("\n ------------------------------------------------------------ \n")
```

```
##
##  ------------------------------------------------------------
```

```
print(p_values)
```

```
##                         symb     normLoss    wheelBase       length
## symb             0.000000e+00 3.432929e-13 2.255894e-16 1.409339e-07
## normLoss         3.432929e-13 0.000000e+00 3.439849e-01 7.679017e-01
## wheelBase        2.255894e-16 3.439849e-01 0.000000e+00 9.699227e-66
## length           1.409339e-07 7.679017e-01 9.699227e-66 0.000000e+00
## width            7.770414e-04 1.805700e-01 5.612752e-46 4.332386e-56
## height           5.531350e-17 7.428895e-09 1.437985e-20 7.731666e-14
## curbWeight       1.025874e-03 1.262266e-01 1.439877e-42 8.728146e-67
## engineSize       1.311313e-01 3.218897e-02 5.233096e-19 1.496334e-29
## bore             5.750916e-02 6.497992e-01 1.475771e-13 1.162914e-21
## stroke           8.994914e-01 4.096535e-01 2.201552e-02 6.640708e-02
## compressionRatio 1.043948e-02 9.039883e-02 3.039092e-04 2.329180e-02
## horsePower       3.098897e-01 1.204793e-04 2.540628e-07 8.505849e-18
## peakRpm          7.359080e-05 6.174988e-04 1.209679e-07 3.232254e-05
## cityMpg          6.101009e-01 8.314261e-04 1.103347e-12 3.595529e-28
## highwayMpg       6.222950e-01 6.748692e-03 3.420917e-17 4.428948e-32
## price            2.449149e-01 9.045759e-03 8.076488e-20 8.016477e-30
##                         width       height   curbWeight   engineSize
## symb             7.770414e-04 5.531350e-17 1.025874e-03 1.311313e-01
## normLoss         1.805700e-01 7.428895e-09 1.262266e-01 3.218897e-02
## wheelBase        5.612752e-46 1.437985e-20 1.439877e-42 5.233096e-19
## length           4.332386e-56 7.731666e-14 8.728146e-67 1.496334e-29
## width            0.000000e+00 5.031446e-05 2.460481e-63 3.685904e-36
## height           5.031446e-05 0.000000e+00 1.688129e-05 3.387566e-01
## curbWeight       2.460481e-63 1.688129e-05 0.000000e+00 1.404581e-58
## engineSize       3.685904e-36 3.387566e-01 1.404581e-58 0.000000e+00
## bore             6.240938e-18 1.234844e-02 2.013256e-25 1.455497e-20
## stroke           9.331725e-03 4.215601e-01 1.651667e-02 3.243224e-03
## compressionRatio 9.348405e-03 1.549235e-04 3.027941e-02 6.800779e-01
## horsePower       5.048681e-25 1.158388e-01 4.321574e-38 1.236008e-48
## peakRpm          1.614542e-03 2.745139e-06 1.228169e-04 4.359121e-04
## cityMpg          2.807249e-25 4.885825e-01 1.929250e-39 2.298962e-26
## highwayMpg       7.321537e-29 1.254830e-01 2.007127e-46 6.864987e-29
## price            9.200336e-38 5.514627e-02 2.189577e-53 9.265492e-64
##                         bore      stroke compressionRatio   horsePower
## symb             5.750916e-02 0.899491367     1.043948e-02 3.098897e-01
## normLoss         6.497992e-01 0.409653480     9.039883e-02 1.204793e-04
## wheelBase        1.475771e-13 0.022015524     3.039092e-04 2.540628e-07
## length           1.162914e-21 0.066407079     2.329180e-02 8.505849e-18
```

```
## width            6.240938e-18 0.009331725     9.348405e-03 5.048681e-25
## height           1.234844e-02 0.421560143     1.549235e-04 1.158388e-01
## curbWeight       2.013256e-25 0.016516670     3.027941e-02 4.321574e-38
## engineSize       1.455497e-20 0.003243224     6.800779e-01 1.236008e-48
## bore             0.000000e+00 0.430509785     9.415682e-01 4.459091e-19
## stroke           4.305098e-01 0.000000000     8.141820e-03 2.048853e-01
## compressionRatio 9.415682e-01 0.008141820     0.000000e+00 3.211079e-03
## horsePower       4.459091e-19 0.204885309     3.211079e-03 0.000000e+00
## peakRpm          1.620994e-04 0.315638096     7.737735e-11 6.252497e-02
## cityMpg          1.328843e-20 0.545323380     2.034939e-06 3.486622e-47
## highwayMpg       1.331729e-20 0.530218077     1.215757e-04 3.062649e-41
## price            1.567358e-16 0.250195842     3.158110e-01 1.189128e-47
##                      peakRpm       cityMpg    highwayMpg        price
## symb             7.359080e-05  6.101009e-01  6.222950e-01 2.449149e-01
## normLoss         6.174988e-04  8.314261e-04  6.748692e-03 9.045759e-03
## wheelBase        1.209679e-07  1.103347e-12  3.420917e-17 8.076488e-20
## length           3.232254e-05  3.595529e-28  4.428948e-32 8.016477e-30
## width            1.614542e-03  2.807249e-25  7.321537e-29 9.200336e-38
## height           2.745139e-06  4.885825e-01  1.254830e-01 5.514627e-02
## curbWeight       1.228169e-04  1.929250e-39  2.007127e-46 2.189577e-53
## engineSize       4.359121e-04  2.298962e-26  6.864987e-29 9.265492e-64
## bore             1.620994e-04  1.328843e-20  1.331729e-20 1.567358e-16
## stroke           3.156381e-01  5.453234e-01  5.302181e-01 2.501958e-01
## compressionRatio 7.737735e-11  2.034939e-06  1.215757e-04 3.158110e-01
## horsePower       6.252497e-02  3.486622e-47  3.062649e-41 1.189128e-47
## peakRpm          0.000000e+00  1.059931e-01  4.419900e-01 1.531182e-01
## cityMpg          1.059931e-01  0.000000e+00 1.248739e-128 2.321132e-29
## highwayMpg       4.419900e-01 1.248739e-128  0.000000e+00 1.749547e-31
## price            1.531182e-01  2.321132e-29  1.749547e-31 0.000000e+00
```

```
cat("\n correlación coeficiente \n")
```

```
##
##  correlación coeficiente
```

```
cat("------------------------------------------------------------ \n")
```

```
## ------------------------------------------------------------
```

```
print(conf_intervals)
```

```
##                       symb    normLoss  wheelBase     length       width
## symb             1.00000000  0.63080959 -0.4259703 -0.2319429 -0.09904538
## normLoss         0.63080959  1.00000000  0.0797976  0.1758437  0.25422871
## wheelBase       -0.42597032  0.07979760  1.0000000  0.9033561  0.84058626
## length          -0.23194294  0.17584372  0.9033561  1.0000000  0.87706091
## width           -0.09904538  0.25422871  0.8405863  0.8770609  1.00000000
## height          -0.43635506 -0.29888785  0.6721760  0.5884675  0.40090603
## curbWeight      -0.09357527  0.26821452  0.8255856  0.9058127  0.89743930
## engineSize       0.03170493  0.31259716  0.6552430  0.7501479  0.79259100
## bore             0.00426889  0.11966331  0.5888129  0.6880306  0.64748909
## stroke           0.12959097  0.21856022  0.2933166  0.2634045  0.31341529
```

```
## compressionRatio -0.042520236  0.02102330  0.3740184  0.2891713  0.31045778
## horsePower        0.207286925  0.42955157  0.4673341  0.6435282  0.71677463
## peakRpm           0.397261736  0.40156389 -0.2350276 -0.1557789 -0.08481670
## cityMpg           0.101711684 -0.10959405 -0.3563500 -0.5879264 -0.55450637
## highwayMpg        0.170830894 -0.05943753 -0.4398407 -0.6282970 -0.59544139
## price             0.056649477  0.34573487  0.6689138  0.7566984  0.80587256
##                       height  curbWeight  engineSize         bore     stroke
## symb             -0.43635506 -0.09357527  0.03170493  0.004268885 0.12959097
## normLoss         -0.29888785  0.26821452  0.31259716  0.119663313 0.21856022
## wheelBase         0.67217597  0.82558555  0.65524297  0.588812949 0.29331657
## length            0.58846753  0.90581274  0.75014787  0.688030613 0.26340450
## width             0.40090603  0.89743930  0.79259100  0.647489085 0.31341529
## height            1.00000000  0.41576666  0.20232204  0.307101599 0.08204266
## curbWeight        0.41576666  1.00000000  0.88452752  0.722538385 0.30030259
## engineSize        0.20232204  0.88452752  1.00000000  0.676836166 0.33547386
## bore              0.30710160  0.72253838  0.67683617  1.000000000 0.08312897
## stroke            0.08204266  0.30030259  0.33547386  0.083128969 1.00000000
## compressionRatio  0.38448636  0.28253642  0.16534992  0.143493926 0.31641259
## horsePower        0.02741710  0.80544148  0.85321930  0.663119197 0.22650041
## peakRpm          -0.19313323 -0.13349205 -0.11063456 -0.129955701 0.06827603
## cityMpg           0.08898852 -0.69222649 -0.56745185 -0.497093339 0.09605909
## highwayMpg        0.03012051 -0.74145660 -0.59574184 -0.497079655 0.09444836
## price             0.26884011  0.87210035  0.90185234  0.634991290 0.21957894
##                  compressionRatio  horsePower     peakRpm     cityMpg
## symb                     -0.04252024  0.20728692  0.39726174  0.10171168
## normLoss                  0.02102330  0.42955157  0.40156389 -0.10959405
## wheelBase                 0.37401843  0.46733415 -0.23502759 -0.35635004
## length                    0.28917132  0.64352816 -0.15577888 -0.58792641
## width                     0.31045778  0.71677463 -0.08481670 -0.55450637
## height                    0.38448636  0.02741710 -0.19313323  0.08898852
## curbWeight                0.28253642  0.80544148 -0.13349205 -0.69222649
## engineSize                0.16534992  0.85321930 -0.11063456 -0.56745185
## bore                      0.14349393  0.66311920 -0.12995570 -0.49709334
## stroke                    0.31641259  0.22650041  0.06827603  0.09605909
## compressionRatio          1.00000000 -0.07015321 -0.31758913  0.44206648
## horsePower               -0.07015321  1.00000000  0.26392057 -0.74877450
## peakRpm                  -0.31758913  0.26392057  1.00000000  0.02430295
## cityMpg                   0.44206648 -0.74877450  0.02430295  1.00000000
## highwayMpg                0.38813099 -0.70840304  0.08408100  0.97817053
## price                     0.20746045  0.85341192  0.03797800 -0.60573157
##                   highwayMpg       price
## symb              0.17083089  0.05664948
## normLoss         -0.05943753  0.34573487
## wheelBase        -0.43984067  0.66891383
## length           -0.62829703  0.75669838
## width            -0.59544139  0.80587256
## height            0.03012051  0.26884011
## curbWeight       -0.74145660  0.87210035
## engineSize       -0.59574184  0.90185234
## bore             -0.49707966  0.63499129
## stroke            0.09444836  0.21957894
## compressionRatio  0.38813099  0.20746045
## horsePower       -0.70840304  0.85341192
## peakRpm           0.08408100  0.03797800
```

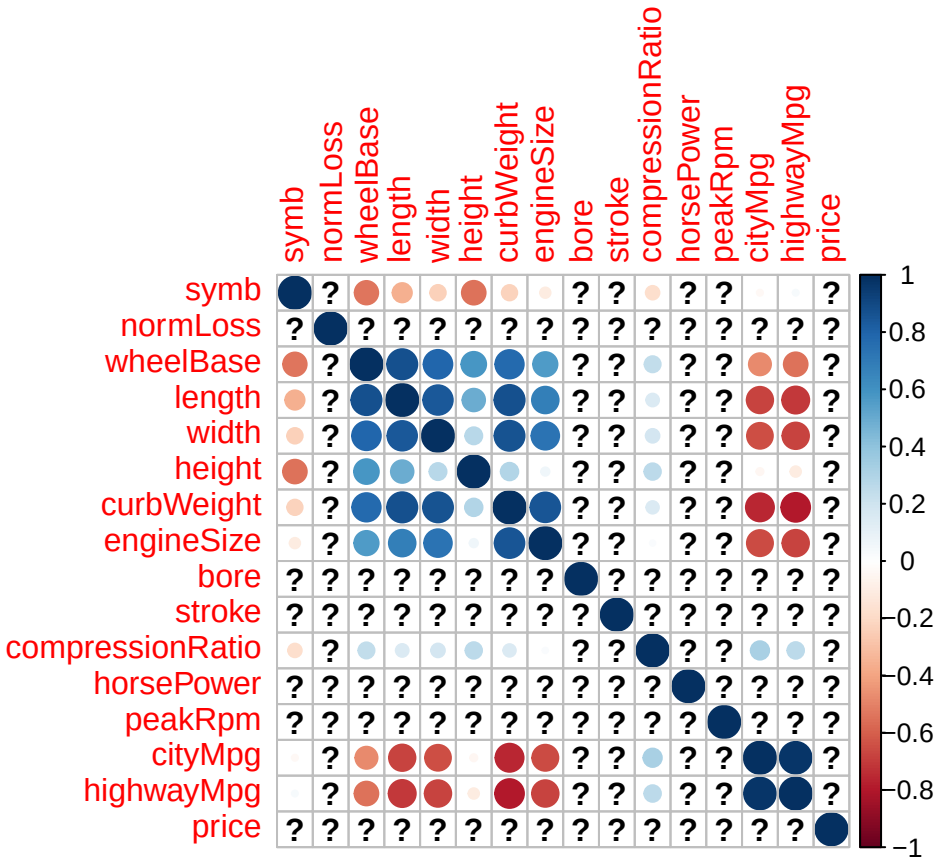
```
## cityMpg            0.97817053 -0.60573157
## highwayMpg         1.00000000 -0.62749437
## price             -0.62749437  1.00000000
```

(c) Plot the all correlation information using the function corrplot. Explore some of its parameters.

```
#install.packages("corrplot")
library(corrplot)

numeric_data <- carIns[, sapply(carIns, is.numeric)]
cor_matrix <- cor(numeric_data)

corrplot(cor_matrix)
```




```
#corrplot(cor_matrix, method = "spearman", type = "lower", tl.cex = 0.8, tl.col = "black", addcolor = "
```

5. Load the data set USJudgeRatings1, from the datasets package, containing lawyers' ratings of state judges in the US Superior Court regarding a set of attributes.

(a) Apply the function prcomp() to obtain the principal components. Inspect how each variable is obtained by the linear combination of each component.

```r
# Cargar el paquete "datasets" que contiene el conjunto de datos
library(datasets)
#USJudgeRatings<- read.csv("C:/Proyectos ML/Mineria/Hands_On_2/data/dataset-29460.csv")

# Cargar el conjunto de datos "USJudgeRatings"
data(USJudgeRatings)
# Obtener solo las columnas numéricas del conjunto de datos
numeric_columns <- USJudgeRatings[, sapply(USJudgeRatings, is.numeric)]

# Aplicar la función prcomp() para obtener los componentes principales
pca <- prcomp(numeric_columns)
# Inspeccionar los resultados
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5    PC6     PC7
## Standard deviation     3.0299 0.9879 0.55470 0.46986 0.26398 0.1741 0.12895
## Proportion of Variance 0.8476 0.0901 0.02841 0.02038 0.00643 0.0028 0.00154
## Cumulative Proportion  0.8476 0.9377 0.96612 0.98650 0.99294 0.9957 0.99727
##                           PC8     PC9    PC10    PC11    PC12
## Standard deviation     0.10868 0.08724 0.07122 0.05571 0.04397
## Proportion of Variance 0.00109 0.00070 0.00047 0.00029 0.00018
## Cumulative Proportion  0.99836 0.99907 0.99953 0.99982 1.00000
```

(b) Load the package ggbiplot and plot the two first components with the function ggbiplot(). You can label each point with the lawyer's name by setting the labels parameter.

```r
library(datasets)
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following objects are masked from 'package:psych':
##
##     %+%, alpha
```

```r
# Load the USJudgeRatings dataset
data(USJudgeRatings)

# Apply PCA using prcomp()
#pca <- prcomp(USJudgeRatings[, 1:12], scale. = TRUE)
pca <- prcomp(USJudgeRatings[,-1],scale. = TRUE)

# Extract the first two principal components
pc1 <- pca$x[, 1]
pc2 <- pca$x[, 2]

# Create a data frame with the PC scores and judge names
df <- data.frame(pc1, pc2, Judge = USJudgeRatings$CONT)
```

```r
# Create a scatter plot of the first two components
ggplot(df, aes(x = pc1, y = pc2, label = Judge)) +
  geom_point() +
  geom_text(hjust = 0, vjust = 0) +
  xlab("Principal Component 1") +
  ylab("Principal Component 2") +
  ggtitle("Principal Components Analysis")
```