INFS_SP5_2023
Predictive Analytics
PRACTICAL 2

Enna H

Contents

1. Missing Values 1
2. Feature Selection 14
3. Correlation analysis 19

1. Missing Values

```
# summary of data
summary(data)
```

```
##       OA            White_British    Low_Occupancy      Unemployed
##  Length:749         Min.   : 7.882   Min.   : 0.000   Min.   : 0.000
##  Class :character   1st Qu.:35.915   1st Qu.: 6.015   1st Qu.: 2.500
##  Mode  :character   Median :44.541   Median :10.000   Median : 4.186
##                     Mean   :44.832   Mean   :11.597   Mean   : 4.510
##                     3rd Qu.:54.472   3rd Qu.:16.107   3rd Qu.: 6.158
##                     Max.   :78.035   Max.   :64.286   Max.   :18.623
##  Qualification
##  Min.   :11.64
##  1st Qu.:36.32
##  Median :55.10
##  Mean   :51.43
##  3rd Qu.:66.23
##  Max.   :88.07
```

```
# check missing values
inspectdf::inspect_na(data)
```

```
## # A tibble: 5 x 3
##   col_name        cnt  pcnt
##   <chr>         <int> <dbl>
## 1 OA                0     0
## 2 White_British     0     0
## 3 Low_Occupancy     0     0
## 4 Unemployed        0     0
## 5 Qualification     0     0
```

- introduce some missing values

```
pacman::p_load(missForest)
```

```
# introduce 10% missing values
data.mis <- prodNA(data, noNA = 0.1)
```

```
# check missing values
inspectdf::inspect_na(data.mis)
```

```
## # A tibble: 5 x 3
##   col_name        cnt  pcnt
##   <chr>         <int> <dbl>
## 1 Low_Occupancy    81 10.8
## 2 OA               78 10.4
## 3 White_British    77 10.3
## 4 Qualification    72  9.61
## 5 Unemployed       66  8.81
```

```
# Remove categorical variables - OA, the first column in this case
data.mis <- subset(data.mis, select = -c(OA))
head(data.mis)
```

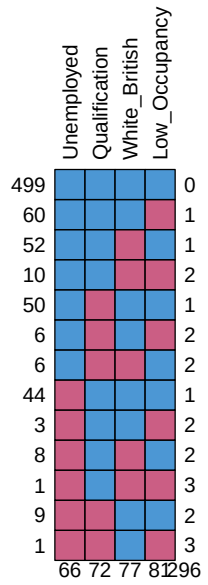
```
##   White_British Low_Occupancy Unemployed Qualification
## 1      42.35669     6.2937063   1.893939      73.62637
## 2      47.20000     5.9322034   2.688172      69.90291
## 3      40.67797     2.9126214         NA      67.58242
## 4      49.66216     0.9259259   2.803738      60.77586
## 5      51.13636     2.0000000   3.816794      65.98639
## 6      41.41791     3.9325843   3.846154      74.20635
```

use mice package

```
# install.packages("mice")
pacman::p_load(mice)
```

- md.pattern() function from the mice package provides a tabular form and a visualisation of the missing data pattern

```
# Explore missing values
md.pattern(data.mis, rotate.names = TRUE)
```



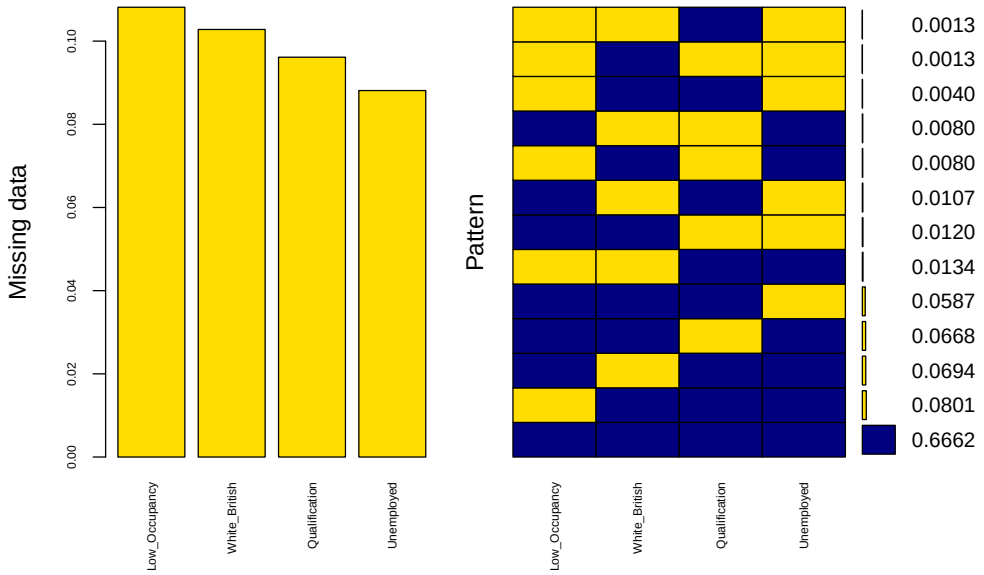

```
##     Unemployed Qualification White_British Low_Occupancy
## 499          1             1             1             1   0
## 60           1             1             1             0   1
## 52           1             1             0             1   1
## 10           1             1             0             0   2
## 50           1             0             1             1   1
## 6            1             0             1             0   2
## 6            1             0             0             1   2
## 44           0             1             1             1   1
## 3            0             1             1             0   2
## 8            0             1             0             1   2
## 1            0             1             0             0   3
## 9            0             0             1             1   2
## 1            0             0             1             0   3
##             66            72            77            81 296
```

- 484 records without any missing values
- 58 observations with missing values in Unemployed
- 49 in Low_Occupancy
- ...

use VIM package

```
# install.packages("VIM")
pacman::p_load(VIM)
```

```
missing.plot <- aggr(data.mis, col=c("navyblue", "#ffdd00"),
                numbers = TRUE, sortVars = TRUE,
                lables=names(data.mis), cex.axis=.6,
                gap=3, ylab=c("Missing data", "Pattern"))
```




```
##
##  Variables sorted by number of missings:
##       Variable      Count
##  Low_Occupancy 0.10814419
##  White_British 0.10280374
##  Qualification 0.09612817
##     Unemployed 0.08811749
```

- impute missing values using mice package

```
# Impute missing values
data.imputed <- mice(data.mis, m=3, maxit = 50, method = 'pmm', seed = 500)
```

```
summary(data.imputed)
```

```
## Class: mids
## Number of multiple imputations:  3
## Imputation methods:
## White_British Low_Occupancy    Unemployed Qualification
##         "pmm"         "pmm"         "pmm"         "pmm"
## PredictorMatrix:
##               White_British Low_Occupancy Unemployed Qualification
## White_British             0             1          1             1
## Low_Occupancy             1             0          1             1
## Unemployed                1             1          0             1
## Qualification             1             1          1             0
```

- m = 3, refers to the umber of imputed datasets. Instead of providing a single value, the method imputes multiple datasets.
- maxit = 50, refers to the number of iterations taken to impute missing values.
- method, refers to the method used in imputation. In this case, we used predictive mean matching.

```
# Explore imputed values
data.imputed$imp$Unemployed
```

```
##              1          2          3
## 3    3.8461538  1.2500000  0.6993007
## 18   2.7472527  1.4218009  2.3121387
## 57   5.7692308  5.5813953  4.4742729
## 62   4.2253521  6.9852941  7.1428571
## 63   4.9180328  7.0588235  4.0123457
## 75   1.7937220  0.0000000  1.0152284
## 89   3.6144578  1.2605042  3.2258065
## 102  0.9523810  1.4925373  2.2026432
## 129  4.2424242  2.7777778  3.5532995
## 159  2.1739130 11.4457831  4.5833333
## 183  1.7391304  2.6881720  0.5208333
## 188  1.6483516  2.7322404  2.4154589
## 190  2.6315789  2.5906736  1.4218009
## 198  3.0456853  4.4843049  3.5000000
## 199  3.7735849  5.1724138  6.3444109
## 207  0.9523810  4.8913043  4.3103448
## 226  1.5957447  3.5087719  4.0485830
## 231  4.5000000  1.8750000  3.2407407
## 233  4.6242775  1.7241379  0.5208333
## 241  2.2058824  4.9019608  6.5116279
## 243  8.5603113  6.8627451  5.0420168
## 268  7.0093458  2.7322404  4.5283019
## 283  2.1978022  4.1666667  2.2624434
## 286  1.7937220  1.6666667  4.6218487
## 287  4.3902439  1.8433180  2.1164021
## 290  3.7313433  4.8913043  4.2553191
## 298  2.5423729  4.0485830  3.7383178
## 302  2.4691358  2.9268293  2.2935780
## 320  3.7735849  2.2598870  4.4742729
## 323  9.0909091 10.6995885  6.1583578
## 335  3.7735849  5.1401869  4.0123457
## 348  4.0909091  2.1739130  2.8571429
## 405  1.2195122  6.5000000  0.0000000
## 423  4.6822742  3.2407407  4.5977011
## 427  4.8913043  2.4154589  1.0869565
## 430  7.4380165  6.3559322  7.2115385
## 444  5.5299539  2.5423729  4.8913043
## 480  8.4745763  1.7937220  6.5989848
## 488  3.8626609  6.4516129  7.4712644
## 501  7.8341014  8.8000000  8.8000000
## 508  8.1196581  9.2592593  5.5555556
## 520  3.6496350  4.3010753  7.4534161
## 536  8.5561497  7.1065990  7.1065990
## 550  6.8825911  3.3434650  0.9090909
## 557  3.8626609  4.0123457  3.6437247
## 563  8.5889571  8.4821429 11.6788321
## 567  4.6979866  0.0000000  7.0093458
## 569  5.9259259  4.0178571  4.5977011
## 599  5.9907834  4.6762590  0.3533569
```

```
## 605  9.9378882 10.1123596  8.9361702
## 607  6.9852941  6.9852941  5.6451613
## 613  3.8888889  6.1224490  3.5294118
## 620 18.6234818  9.9378882 11.6788321
## 625  8.5889571  6.4267352  8.8000000
## 633  3.3519553  1.4925373  1.4925373
## 644  4.5714286  3.8461538  1.0526316
## 651  5.6818182  1.0204082  1.4218009
## 654 10.5882353  8.4821429  8.4821429
## 662  2.6490066  6.0465116 18.6234818
## 666  2.0618557  1.3953488  2.6881720
## 673  2.6086957  1.7793594  1.6853933
## 719  5.3571429  5.1401869  5.8333333
## 725  1.0204082  0.8130081  2.8708134
## 731  3.5087719  2.7322404  3.8251366
## 742  0.5847953  0.9523810  1.4084507
## 749  6.0483871  6.0483871  6.0483871
```

since we set m=3, you should see 3 columns, one for each of the imputed datasets.

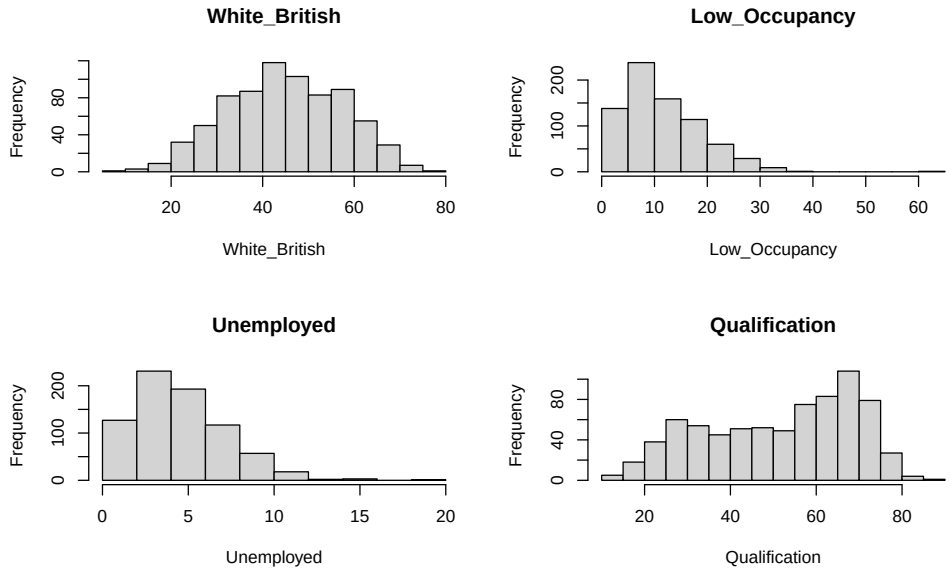
- use complete() function to extract the completed data (second dataset)

```
# Select second complete dataset (out of 3)
data.complete <- complete(data.imputed, 2)
head(data.complete)
```

```
##   White_British Low_Occupancy Unemployed Qualification
## 1      42.35669     6.2937063   1.893939      73.62637
## 2      47.20000     5.9322034   2.688172      69.90291
## 3      40.67797     2.9126214   1.250000      67.58242
## 4      49.66216     0.9259259   2.803738      60.77586
## 5      51.13636     2.0000000   3.816794      65.98639
## 6      41.41791     3.9325843   3.846154      74.20635
```

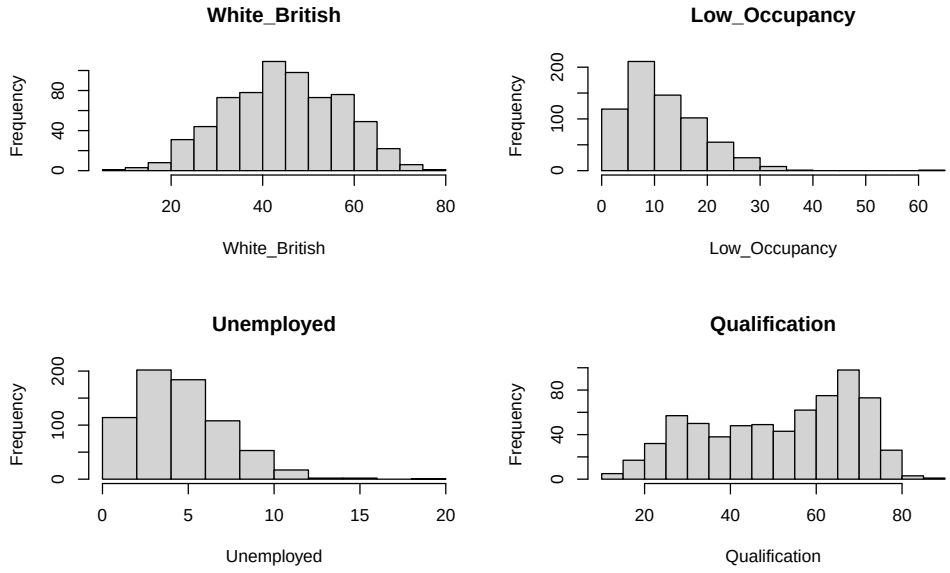
- plot histograms for each of the four columns (White_British, Low_Occupancy, Unemployed, Qualification) for the original dataset

```
# Plot histograms for each of the four columns (White_British, Low_Occupancy, Unemployed, Qualification
par(mfrow=c(2,2))
hist(data$White_British, main="White_British", xlab="White_British")
hist(data$Low_Occupancy, main="Low_Occupancy", xlab="Low_Occupancy")
hist(data$Unemployed, main="Unemployed", xlab="Unemployed")
hist(data$Qualification, main="Qualification", xlab="Qualification")
```



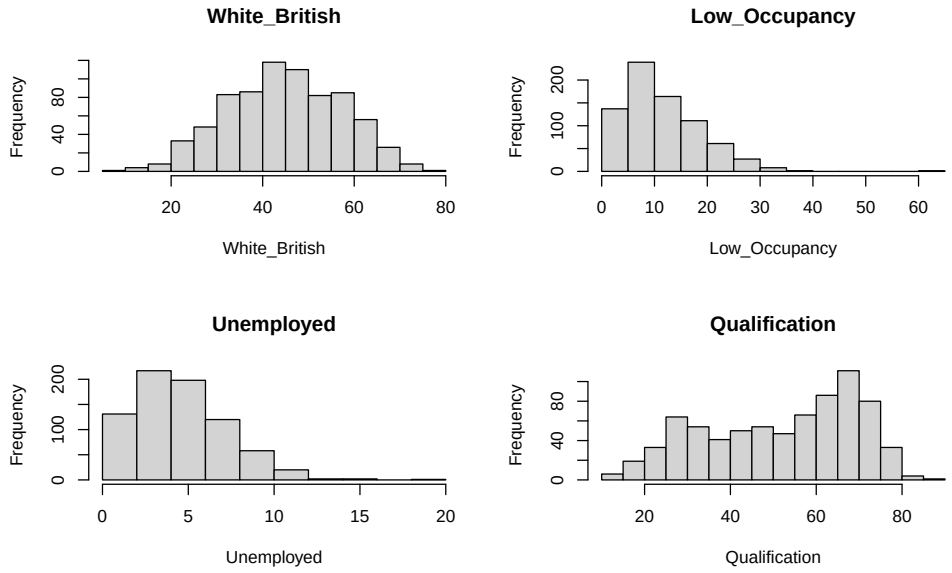
- before imputation

```
# Plot histograms for each of the four columns (White_British, Low_Occupancy, Unemployed, Qualification
par(mfrow=c(2,2))
hist(data.mis$White_British, main="White_British", xlab="White_British")
hist(data.mis$Low_Occupancy, main="Low_Occupancy", xlab="Low_Occupancy")
hist(data.mis$Unemployed, main="Unemployed", xlab="Unemployed")
hist(data.mis$Qualification, main="Qualification", xlab="Qualification")
```



- plot histograms for each of the four columns (White_British, Low_Occupancy, Unemployed, Qualification) for the imputed dataset (second dataset)

```
par(mfrow=c(2,2))
hist(data.complete$White_British, main="White_British", xlab="White_British")
hist(data.complete$Low_Occupancy, main="Low_Occupancy", xlab="Low_Occupancy")
hist(data.complete$Unemployed, main="Unemployed", xlab="Unemployed")
hist(data.complete$Qualification, main="Qualification", xlab="Qualification")
```



- t test to compare the means of the original and imputed datasets

```
t_test_result_1 <- t.test(data$White_British, data.complete$White_British)
t_test_result_1
```

```
##
##  Welch Two Sample t-test
##
## data:  data$White_British and data.complete$White_British
## t = 0.10025, df = 1496, p-value = 0.9202
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.204050  1.333748
## sample estimates:
## mean of x mean of y
##  44.83223  44.76738
```

```
columns_to_test <- c("White_British", "Low_Occupancy", "Unemployed", "Qualification")

t_test_results <- sapply(columns_to_test, function(col_name) {
  t.test(data[[col_name]], data.complete[[col_name]])$p.value
})

names(t_test_results) <- columns_to_test
t_test_results
```

```
## White_British Low_Occupancy    Unemployed Qualification
##     0.9201603     0.9373360     0.8324054     0.7412118
```

The p-values are greater than the typical 0.05 significance level, indicating that there's no significant difference in means between the original and imputed data for these variables.

- No Significant Difference: For all four variables, the p-values suggest that there is no statistically significant difference in means between the original dataset and the imputed dataset. This means that the imputation process did not significantly alter the central tendency of these variables.
- Impact of Imputation: Since the t-tests did not detect significant differences, it could be inferred that the imputation method used was conservative in the sense that it did not significantly shift the means of these variables. It is consistent with an imputation process that maintains the overall distribution of the original data.

In the context of data imputation, these results can be seen as favorable, especially if the goal was to fill missing values in a way that preserves the overall statistical properties of the dataset. It might be beneficial to also assess the quality of the imputation in terms of the specific missing data mechanism (e.g., Missing Completely At Random, Missing At Random, Not Missing At Random) and the chosen imputation technique, as the suitability and impact of imputation can vary based on these factors.

Challenge 1. In the example above, we used pmm - Predictive Mean Matching as a method to impute data. As you can see in the documentation for mice, there are other methods available.

- Choose two (2) additional methods (for instance, cart or quadratic) and repeat the imputation procedure from above.

cart

```
# Impute missing values with cart
data.imputed.cart <- mice(data.mis, m=3, maxit = 50, method = 'cart', seed = 500)
```

```
summary(data.imputed.cart)
```

```
## Class: mids
## Number of multiple imputations:  3
## Imputation methods:
## White_British Low_Occupancy    Unemployed Qualification
##        "cart"        "cart"        "cart"        "cart"
## PredictorMatrix:
##               White_British Low_Occupancy Unemployed Qualification
## White_British             0             1          1             1
## Low_Occupancy             1             0          1             1
## Unemployed                1             1          0             1
## Qualification             1             1          1             0
```

use complete() function to extract the completed data (second dataset)

```
# Select second complete dataset (out of 3)
data.complete.cart <- complete(data.imputed.cart, 2)
head(data.complete.cart)
```

```
##   White_British Low_Occupancy Unemployed Qualification
## 1      42.35669     6.2937063   1.893939      73.62637
## 2      47.20000     5.9322034   2.688172      69.90291
## 3      40.67797     2.9126214   2.127660      67.58242
## 4      49.66216     0.9259259   2.803738      60.77586
## 5      51.13636     2.0000000   3.816794      65.98639
## 6      41.41791     3.9325843   3.846154      74.20635
```

- t test to compare the means of the original and imputed datasets

```r
columns_to_test <- c("White_British", "Low_Occupancy", "Unemployed", "Qualification")

t_test_results_cart <- sapply(columns_to_test, function(col_name) {
  t.test(data[[col_name]], data.complete.cart[[col_name]])$p.value
})

names(t_test_results_cart) <- columns_to_test
t_test_results_cart
```

```
## White_British Low_Occupancy    Unemployed Qualification
##     0.6226115     0.8679482     0.6146239     0.9649831
```

norm.predict

```r
# Impute missing values with norm.predict
data.imputed.norm <- mice(data.mis, m=3, maxit = 50, method = 'norm.predict', seed = 500)
```

```
##
##  iter imp variable
##   1   1  White_British  Low_Occupancy  Unemployed  Qualification
##   1   2  White_British  Low_Occupancy  Unemployed  Qualification
##   1   3  White_British  Low_Occupancy  Unemployed  Qualification
##   2   1  White_British  Low_Occupancy  Unemployed  Qualification
##   2   2  White_British  Low_Occupancy  Unemployed  Qualification
##   2   3  White_British  Low_Occupancy  Unemployed  Qualification
##   3   1  White_British  Low_Occupancy  Unemployed  Qualification
##   3   2  White_British  Low_Occupancy  Unemployed  Qualification
##   3   3  White_British  Low_Occupancy  Unemployed  Qualification
##   4   1  White_British  Low_Occupancy  Unemployed  Qualification
##   4   2  White_British  Low_Occupancy  Unemployed  Qualification
##   4   3  White_British  Low_Occupancy  Unemployed  Qualification
##   5   1  White_British  Low_Occupancy  Unemployed  Qualification
##   5   2  White_British  Low_Occupancy  Unemployed  Qualification
##   5   3  White_British  Low_Occupancy  Unemployed  Qualification
##   6   1  White_British  Low_Occupancy  Unemployed  Qualification
##   6   2  White_British  Low_Occupancy  Unemployed  Qualification
##   6   3  White_British  Low_Occupancy  Unemployed  Qualification
##   7   1  White_British  Low_Occupancy  Unemployed  Qualification
##   7   2  White_British  Low_Occupancy  Unemployed  Qualification
##   7   3  White_British  Low_Occupancy  Unemployed  Qualification
##   8   1  White_British  Low_Occupancy  Unemployed  Qualification
##   8   2  White_British  Low_Occupancy  Unemployed  Qualification
```

```
##   8   3  White_British  Low_Occupancy  Unemployed  Qualification
##   9   1  White_British  Low_Occupancy  Unemployed  Qualification
##   9   2  White_British  Low_Occupancy  Unemployed  Qualification
##   9   3  White_British  Low_Occupancy  Unemployed  Qualification
##   10   1  White_British  Low_Occupancy  Unemployed  Qualification
##   10   2  White_British  Low_Occupancy  Unemployed  Qualification
##   10   3  White_British  Low_Occupancy  Unemployed  Qualification
##   11   1  White_British  Low_Occupancy  Unemployed  Qualification
##   11   2  White_British  Low_Occupancy  Unemployed  Qualification
##   11   3  White_British  Low_Occupancy  Unemployed  Qualification
##   12   1  White_British  Low_Occupancy  Unemployed  Qualification
##   12   2  White_British  Low_Occupancy  Unemployed  Qualification
##   12   3  White_British  Low_Occupancy  Unemployed  Qualification
##   13   1  White_British  Low_Occupancy  Unemployed  Qualification
##   13   2  White_British  Low_Occupancy  Unemployed  Qualification
##   13   3  White_British  Low_Occupancy  Unemployed  Qualification
##   14   1  White_British  Low_Occupancy  Unemployed  Qualification
##   14   2  White_British  Low_Occupancy  Unemployed  Qualification
##   14   3  White_British  Low_Occupancy  Unemployed  Qualification
##   15   1  White_British  Low_Occupancy  Unemployed  Qualification
##   15   2  White_British  Low_Occupancy  Unemployed  Qualification
##   15   3  White_British  Low_Occupancy  Unemployed  Qualification
##   16   1  White_British  Low_Occupancy  Unemployed  Qualification
##   16   2  White_British  Low_Occupancy  Unemployed  Qualification
##   16   3  White_British  Low_Occupancy  Unemployed  Qualification
##   17   1  White_British  Low_Occupancy  Unemployed  Qualification
##   17   2  White_British  Low_Occupancy  Unemployed  Qualification
##   17   3  White_British  Low_Occupancy  Unemployed  Qualification
##   18   1  White_British  Low_Occupancy  Unemployed  Qualification
##   18   2  White_British  Low_Occupancy  Unemployed  Qualification
##   18   3  White_British  Low_Occupancy  Unemployed  Qualification
##   19   1  White_British  Low_Occupancy  Unemployed  Qualification
##   19   2  White_British  Low_Occupancy  Unemployed  Qualification
##   19   3  White_British  Low_Occupancy  Unemployed  Qualification
##   20   1  White_British  Low_Occupancy  Unemployed  Qualification
##   20   2  White_British  Low_Occupancy  Unemployed  Qualification
##   20   3  White_British  Low_Occupancy  Unemployed  Qualification
##   21   1  White_British  Low_Occupancy  Unemployed  Qualification
##   21   2  White_British  Low_Occupancy  Unemployed  Qualification
##   21   3  White_British  Low_Occupancy  Unemployed  Qualification
##   22   1  White_British  Low_Occupancy  Unemployed  Qualification
##   22   2  White_British  Low_Occupancy  Unemployed  Qualification
##   22   3  White_British  Low_Occupancy  Unemployed  Qualification
##   23   1  White_British  Low_Occupancy  Unemployed  Qualification
##   23   2  White_British  Low_Occupancy  Unemployed  Qualification
##   23   3  White_British  Low_Occupancy  Unemployed  Qualification
##   24   1  White_British  Low_Occupancy  Unemployed  Qualification
##   24   2  White_British  Low_Occupancy  Unemployed  Qualification
##   24   3  White_British  Low_Occupancy  Unemployed  Qualification
##   25   1  White_British  Low_Occupancy  Unemployed  Qualification
##   25   2  White_British  Low_Occupancy  Unemployed  Qualification
##   25   3  White_British  Low_Occupancy  Unemployed  Qualification
##   26   1  White_British  Low_Occupancy  Unemployed  Qualification
##   26   2  White_British  Low_Occupancy  Unemployed  Qualification
```

```
##   26   3  White_British  Low_Occupancy  Unemployed  Qualification
##   27   1  White_British  Low_Occupancy  Unemployed  Qualification
##   27   2  White_British  Low_Occupancy  Unemployed  Qualification
##   27   3  White_British  Low_Occupancy  Unemployed  Qualification
##   28   1  White_British  Low_Occupancy  Unemployed  Qualification
##   28   2  White_British  Low_Occupancy  Unemployed  Qualification
##   28   3  White_British  Low_Occupancy  Unemployed  Qualification
##   29   1  White_British  Low_Occupancy  Unemployed  Qualification
##   29   2  White_British  Low_Occupancy  Unemployed  Qualification
##   29   3  White_British  Low_Occupancy  Unemployed  Qualification
##   30   1  White_British  Low_Occupancy  Unemployed  Qualification
##   30   2  White_British  Low_Occupancy  Unemployed  Qualification
##   30   3  White_British  Low_Occupancy  Unemployed  Qualification
##   31   1  White_British  Low_Occupancy  Unemployed  Qualification
##   31   2  White_British  Low_Occupancy  Unemployed  Qualification
##   31   3  White_British  Low_Occupancy  Unemployed  Qualification
##   32   1  White_British  Low_Occupancy  Unemployed  Qualification
##   32   2  White_British  Low_Occupancy  Unemployed  Qualification
##   32   3  White_British  Low_Occupancy  Unemployed  Qualification
##   33   1  White_British  Low_Occupancy  Unemployed  Qualification
##   33   2  White_British  Low_Occupancy  Unemployed  Qualification
##   33   3  White_British  Low_Occupancy  Unemployed  Qualification
##   34   1  White_British  Low_Occupancy  Unemployed  Qualification
##   34   2  White_British  Low_Occupancy  Unemployed  Qualification
##   34   3  White_British  Low_Occupancy  Unemployed  Qualification
##   35   1  White_British  Low_Occupancy  Unemployed  Qualification
##   35   2  White_British  Low_Occupancy  Unemployed  Qualification
##   35   3  White_British  Low_Occupancy  Unemployed  Qualification
##   36   1  White_British  Low_Occupancy  Unemployed  Qualification
##   36   2  White_British  Low_Occupancy  Unemployed  Qualification
##   36   3  White_British  Low_Occupancy  Unemployed  Qualification
##   37   1  White_British  Low_Occupancy  Unemployed  Qualification
##   37   2  White_British  Low_Occupancy  Unemployed  Qualification
##   37   3  White_British  Low_Occupancy  Unemployed  Qualification
##   38   1  White_British  Low_Occupancy  Unemployed  Qualification
##   38   2  White_British  Low_Occupancy  Unemployed  Qualification
##   38   3  White_British  Low_Occupancy  Unemployed  Qualification
##   39   1  White_British  Low_Occupancy  Unemployed  Qualification
##   39   2  White_British  Low_Occupancy  Unemployed  Qualification
##   39   3  White_British  Low_Occupancy  Unemployed  Qualification
##   40   1  White_British  Low_Occupancy  Unemployed  Qualification
##   40   2  White_British  Low_Occupancy  Unemployed  Qualification
##   40   3  White_British  Low_Occupancy  Unemployed  Qualification
##   41   1  White_British  Low_Occupancy  Unemployed  Qualification
##   41   2  White_British  Low_Occupancy  Unemployed  Qualification
##   41   3  White_British  Low_Occupancy  Unemployed  Qualification
##   42   1  White_British  Low_Occupancy  Unemployed  Qualification
##   42   2  White_British  Low_Occupancy  Unemployed  Qualification
##   42   3  White_British  Low_Occupancy  Unemployed  Qualification
##   43   1  White_British  Low_Occupancy  Unemployed  Qualification
##   43   2  White_British  Low_Occupancy  Unemployed  Qualification
##   43   3  White_British  Low_Occupancy  Unemployed  Qualification
##   44   1  White_British  Low_Occupancy  Unemployed  Qualification
##   44   2  White_British  Low_Occupancy  Unemployed  Qualification
```

```
##   44   3  White_British  Low_Occupancy  Unemployed  Qualification
##   45   1  White_British  Low_Occupancy  Unemployed  Qualification
##   45   2  White_British  Low_Occupancy  Unemployed  Qualification
##   45   3  White_British  Low_Occupancy  Unemployed  Qualification
##   46   1  White_British  Low_Occupancy  Unemployed  Qualification
##   46   2  White_British  Low_Occupancy  Unemployed  Qualification
##   46   3  White_British  Low_Occupancy  Unemployed  Qualification
##   47   1  White_British  Low_Occupancy  Unemployed  Qualification
##   47   2  White_British  Low_Occupancy  Unemployed  Qualification
##   47   3  White_British  Low_Occupancy  Unemployed  Qualification
##   48   1  White_British  Low_Occupancy  Unemployed  Qualification
##   48   2  White_British  Low_Occupancy  Unemployed  Qualification
##   48   3  White_British  Low_Occupancy  Unemployed  Qualification
##   49   1  White_British  Low_Occupancy  Unemployed  Qualification
##   49   2  White_British  Low_Occupancy  Unemployed  Qualification
##   49   3  White_British  Low_Occupancy  Unemployed  Qualification
##   50   1  White_British  Low_Occupancy  Unemployed  Qualification
##   50   2  White_British  Low_Occupancy  Unemployed  Qualification
##   50   3  White_British  Low_Occupancy  Unemployed  Qualification
```

```
summary(data.imputed.norm)
```

```
## Class: mids
## Number of multiple imputations:  3
## Imputation methods:
##  White_British  Low_Occupancy     Unemployed  Qualification
## "norm.predict" "norm.predict" "norm.predict" "norm.predict"
## PredictorMatrix:
##               White_British Low_Occupancy Unemployed Qualification
## White_British             0             1          1             1
## Low_Occupancy             1             0          1             1
## Unemployed                1             1          0             1
## Qualification             1             1          1             0
```

use complete() function to extract the completed data (second dataset)

```
# Select second complete dataset (out of 3)
data.complete.norm <- complete(data.imputed.norm, 2)
head(data.complete.norm)
```

```
##   White_British Low_Occupancy Unemployed Qualification
## 1      42.35669     6.2937063   1.893939      73.62637
## 2      47.20000     5.9322034   2.688172      69.90291
## 3      40.67797     2.9126214   2.249288      67.58242
## 4      49.66216     0.9259259   2.803738      60.77586
## 5      51.13636     2.0000000   3.816794      65.98639
## 6      41.41791     3.9325843   3.846154      74.20635
```

- t test to compare the means of the original and imputed datasets

```r
columns_to_test <- c("White_British", "Low_Occupancy", "Unemployed", "Qualification")

t_test_results_norm <- sapply(columns_to_test, function(col_name) {
  t.test(data[[col_name]], data.complete.norm[[col_name]])$p.value
})

names(t_test_results_norm) <- columns_to_test
t_test_results_norm
```

```
## White_British Low_Occupancy    Unemployed Qualification
##     0.7066679     0.9839796     0.7656022     0.8860455
```

- explain how the methods selected are different from pmm? any differences in imputed data?

All three methods result in p-values that are not statistically significant, indicating that the imputed datasets do not have a statistically different mean from the original dataset for the tested variables. This is a good sign, as it suggests that the imputation methods have maintained essential statistical properties of the data.

However, these methods differ in their underlying assumptions and how they generate imputations:

- pmm preserves the original distribution and is more flexible.
- cart can capture complex relationships but might overfit if there's noise.
- norm.predict assumes linear relationships and could be inappropriate if the true relationship is nonlinear.

The choice between these methods depends on the underlying relationships between variables in the data, and it might be beneficial to consider diagnostic plots or other model evaluation criteria to select the most appropriate method for your particular dataset.

2. Feature Selection

- Brute-force approach, where we try all possible feature subsets as input to data mining algorithm. This approach is computationally expensive and not feasible for large datasets.
- Embedded approaches, where feature selection occurs naturally as part of the data mining algorithm. For example, decision trees and neural networks automatically select features as part of their algorithm.
- Filter approaches, where we use a statistical measure to score the relevance of each feature. This is a fast and efficient approach, but it does not take into account the interaction between features.
- Wrapper approaches, where we use a data mining algorithm to score the relevance of each feature, black-box style. This approach is computationally expensive, but it can take into account the interaction between features.

Filter approaches are used here, mlr3 package is used to implement the filter approach.

```r
# install.packages("mlr3")
pacman::p_load(mlr3, mlr3learners, mlr3viz, mlr3filters, FSelectorRcpp, mlr3pipelines)
```

```r
set.seed(999)
```

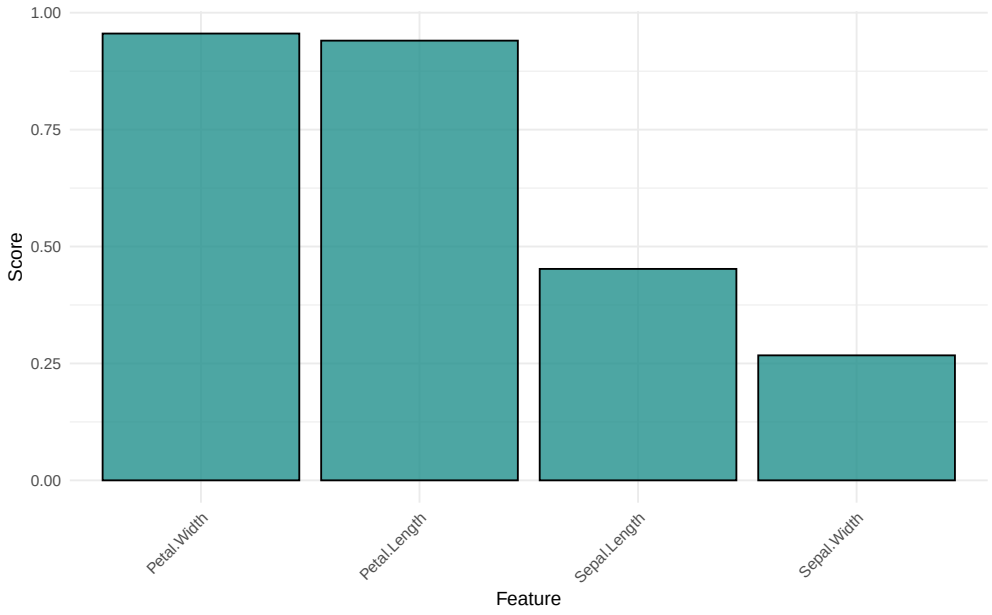
```r
filter.importance = flt("information_gain") # Creating a new filter
iris.task <- tsk("iris") # Creating a Task() object from the Iris dataset
iris.feature.importance <- filter.importance$calculate(iris.task) # Calculating the feature importance
as.data.table(iris.feature.importance) # Converting the feature importance to a data.table
```

```
##         feature     score
## 1:  Petal.Width 0.9554360
## 2: Petal.Length 0.9402853
## 3: Sepal.Length 0.4521286
## 4:  Sepal.Width 0.2672750
```

- Petal.Width has the highest predictive power
- Sepal.Width can likely be ignored for the classification task

plot feature importance

```
autoplot(iris.feature.importance)
```



use mlr3pipelines package to create a pipeline

- filter.nfeat - number of features to select. Mutually exclusive with frac and cutoff,
- filter.frac - fraction of features to keep. Mutually exclusive with nfeat and cutoff, or
- filter.cutoff - minimum value of filter heuristic for which to keep features. Mutually exclusive with nfeat and frac.

```
po = po("filter", filter.importance, filter.nfeat=3) # Creating a pipeline
filtered.task = po$train(list(iris.task))[[1]] # Applying the pipeline to the task
filtered.task$feature_names # Printing the names of the selected features
```

```
## [1] "Petal.Length" "Petal.Width"  "Sepal.Length"
```

set number of features to return to 3 which means that our pipeline should return the list of the three most important features for this specific task.

```
iris.filtered = subset(iris, select = filtered.task$feature_names)
head(iris.filtered)
```

```
##   Petal.Length Petal.Width Sepal.Length
## 1          1.4         0.2          5.1
## 2          1.4         0.2          4.9
## 3          1.3         0.2          4.7
## 4          1.5         0.2          4.6
## 5          1.4         0.2          5.0
## 6          1.7         0.4          5.4
```

Challenge 2. In the example above, we used information_gain as a method to select relevant features. there are many other implemented filters.

- Choose three (3) additional and relevant methods and repeat the feature selection process from above.
- mlr_filters

```
filter.importance = flt("information_gain") # Creating a new filter
iris.task <- tsk("iris") # Creating a Task() object from the Iris dataset
iris.feature.importance <- filter.importance$calculate(iris.task) # Calculating the feature importance
information_gain <- as.data.table(iris.feature.importance) # Converting the feature importance to a dat
```

```
filter.importance = flt("disr") # Creating a new filter
iris.task <- tsk("iris") # Creating a Task() object from the Iris dataset
iris.feature.importance <- filter.importance$calculate(iris.task) # Calculating the feature importance
disr <- as.data.table(iris.feature.importance) # Converting the feature importance to a data.table
```

```
filter.importance = flt("kruskal_test") # Creating a new filter
iris.task <- tsk("iris") # Creating a Task() object from the Iris dataset
iris.feature.importance <- filter.importance$calculate(iris.task) # Calculating the feature importance
kruskal_test <- as.data.table(iris.feature.importance) # Converting the feature importance to a data.ta
```

```
# Verifying each data frame
print("Information Gain:")
```

```
## [1] "Information Gain:"
```

```
print(information_gain)
```

```
##         feature     score
## 1:  Petal.Width 0.9554360
## 2: Petal.Length 0.9402853
## 3: Sepal.Length 0.4521286
## 4:  Sepal.Width 0.2672750
```

```
print("Relief:")
```

```
## [1] "Relief:"
```

```
print(relief)
```

```
##         feature     score
## 1:  Petal.Width 0.3050000
## 2: Petal.Length 0.2962712
## 3: Sepal.Length 0.1419444
## 4:  Sepal.Width 0.1079167
```

```
print("DISR:")
```

```
## [1] "DISR:"
```

```
print(disr)
```

```
##         feature     score
## 1:  Petal.Width 1.0000000
## 2: Petal.Length 0.6666667
## 3: Sepal.Length 0.3333333
## 4:  Sepal.Width 0.0000000
```

```
print("Kruskal Test:")
```

```
## [1] "Kruskal Test:"
```

```
print(kruskal_test)
```

```
##         feature    score
## 1:  Petal.Width 28.48654
## 2: Petal.Length 28.31840
## 3: Sepal.Length 21.04970
## 4:  Sepal.Width 13.80430
```

- Can you explain how the methods you selected are different from information_gain? Can you observe any differences in the feature importance?

```
# Combine the data frames
results <- data.frame(
  feature = c(rep("Petal.Width", 4), rep("Petal.Length", 4), rep("Sepal.Length", 4), rep("Sepal.Width",
  method = rep(c("Information Gain", "Relief", "DISR", "Kruskal Test"), 4),
  score = c(information_gain$score, relief$score, disr$score, kruskal_test$score)
)

# Print the table
kable(results)
```

feature	method	score
Petal.Width	Information Gain	0.9554360

feature	method	score
Petal.Width	Relief	0.9402853
Petal.Width	DISR	0.4521286
Petal.Width	Kruskal Test	0.2672750
Petal.Length	Information Gain	0.3050000
Petal.Length	Relief	0.2962712
Petal.Length	DISR	0.1419444
Petal.Length	Kruskal Test	0.1079167
Sepal.Length	Information Gain	1.0000000
Sepal.Length	Relief	0.6666667
Sepal.Length	DISR	0.3333333
Sepal.Length	Kruskal Test	0.0000000
Sepal.Width	Information Gain	28.4865433
Sepal.Width	Relief	28.3183994
Sepal.Width	DISR	21.0496968
Sepal.Width	Kruskal Test	13.8042990

Information Gain: The "Information Gain" method measures the reduction in uncertainty about the class labels using entropy. It focuses on the ability of features to split the data into subsets that are more homogeneous with respect to the target variable. This method assigns the highest importance scores to "Petal.Width" and "Sepal.Length," indicating their strong predictive power for classifying the Iris dataset.

Relief: The "Relief" method is a filter feature selection technique that focuses on distinguishing instances that are close to each other from those that are far apart. It calculates feature importance based on how well features can separate instances of different classes. Unlike "information_gain," which uses entropy to measure the reduction in uncertainty about the class labels, "Relief" evaluates the ability of features to differentiate between instances. The results show that "Relief" assigns lower scores to "Petal.Width" and "Petal.Length" compared to "information_gain," indicating that these features might have less discriminative power according to this method.

DISR (Distance-based Similarity-Weighted Relief): "DISR" is another variant of the Relief algorithm that incorporates distance-based similarity weighting. It considers the similarity between instances when updating feature weights. This method aims to mitigate the limitation of "Relief," where it may assign equal weights to irrelevant and relevant instances. The scores assigned by "DISR" are generally lower than those of "information_gain," indicating a more conservative approach to feature importance. This could suggest that "DISR" is giving less importance to certain features compared to "information_gain."

Kruskal Test: The "Kruskal Test" is a statistical method used to compare the distribution of a numeric variable across different groups. In the context of feature selection, it evaluates whether the means of a feature vary significantly across different classes. This method is particularly suitable for situations where the classes are categorical, as in the case of the Iris dataset. The "Kruskal Test" assigns significantly lower scores to most features compared to "information_gain." This might suggest that the class-related differences captured by this test are not as pronounced as the entropy-based importance calculated by "information_gain."

In summary, the selected methods differ in their underlying mechanisms for calculating feature importance. "Relief" and "DISR" emphasize instance separability and similarity, respectively, while the "Kruskal Test" leverages statistical significance across class distributions. These differences in methodology result in variations in the ranking and scores of feature importance compared to the "information_gain" method.

1. **Consistency of High Importance:**
    - For the "Information Gain" method, the feature "Sepal.Length" consistently receives the highest importance score (1.00) among all features. This suggests that "Sepal.Length" is deemed highly relevant for classifying the Iris dataset according to this method.
2. **Comparative Importance Scores:**

- Across different methods, we see variations in the importance scores assigned to each feature. Notably, "Petal.Width" consistently ranks high in importance across "Information Gain," "Relief," and "DISR" methods. This consistency might indicate that "Petal.Width" has a significant impact on class separation and prediction across different methodologies.
- "Sepal.Width" exhibits very high importance scores in comparison to other features, particularly in the "Information Gain" and "Relief" methods. However, its importance is relatively lower in the "DISR" and "Kruskal Test" methods. This discrepancy suggests that "Sepal.Width" might play a more prominent role in these particular methods, potentially indicating its significance in some specific contexts.

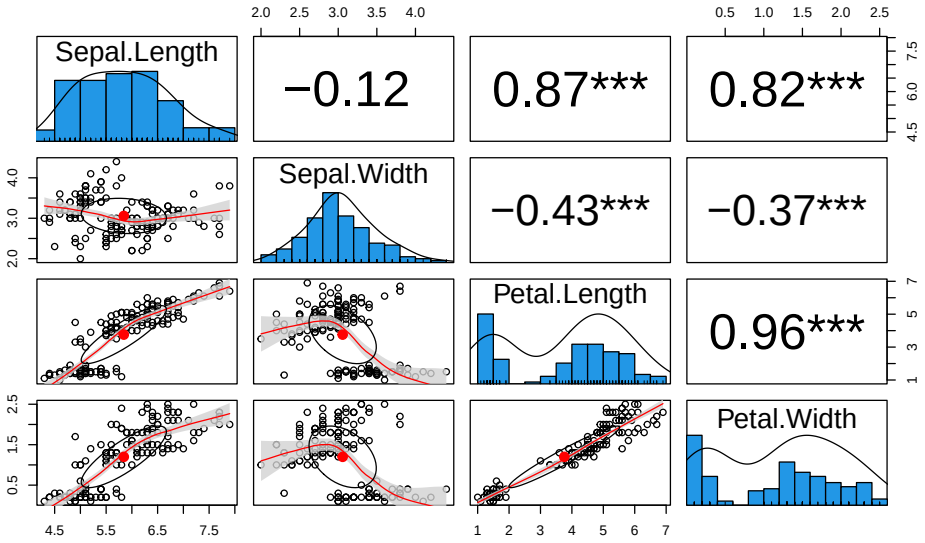
3. **Differential Impact of Methods:**
   - The "Kruskal Test" assigns lower scores to almost all features, implying that the class-related differences captured by this test are not as pronounced in this dataset compared to the other methods. This may indicate that class distributions might not significantly vary for these features.
4. **Feature Discrimination:**
   - The lower importance scores for "Petal.Length" in various methods, especially "DISR" and "Kruskal Test," could imply that it might not have as strong discriminatory power as "Petal.Width" and "Sepal.Length" in this context.
5. **Significance of Sepal.Length and Sepal.Width:**
   - The "Kruskal Test" assigns a score of 0 to "Sepal.Length," which indicates that this feature might not exhibit significant class-based variations according to this method. Similarly, "Sepal.Width" receives comparatively lower scores in "DISR" and "Kruskal Test." This suggests that while "Sepal.Width" might have high importance according to "Information Gain" and "Relief," its significance could be challenged by the underlying characteristics assessed by "DISR" and "Kruskal Test."

3. Correlation analysis

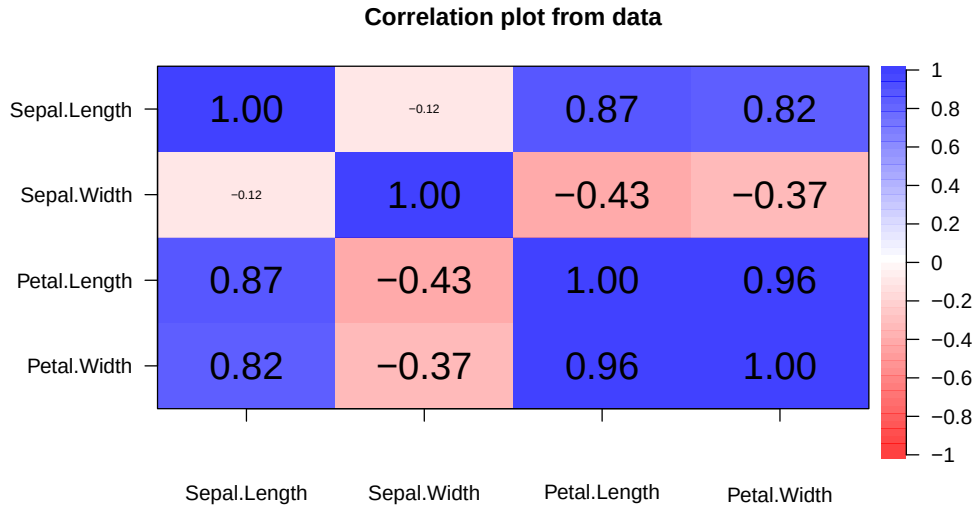
For the purpose of building predictive models, you will want to avoid using highly correlated features. This is because highly correlated features can cause problems when fitting models, such as multicollinearity in linear regression, which can lead to unstable parameter estimates.

```
# Install and load the car package
pacman::p_load(psych, corrplot)
```

```
# install.packages("psych")
library(psych)
data <- iris[, 1:4] # Numerical variables
pairs.panels(data,
             smooth = TRUE,      # If TRUE, draws loess smooths
             scale = FALSE,      # If TRUE, scales the correlation text font
             density = TRUE,     # If TRUE, adds density plots and histograms
             ellipses = TRUE,    # If TRUE, draws ellipses
             method = "pearson", # Correlation method (also "spearman" or "kendall")
             pch = 21,           # pch symbol
             lm = FALSE,         # If TRUE, plots linear fit rather than the LOESS (smoothed) fit
             cor = TRUE,         # If TRUE, reports correlations
             jiggle = FALSE,     # If TRUE, data points are jittered
             factor = 2,         # Jittering factor
             hist.col = 4,       # Histograms color
             stars = TRUE,       # If TRUE, adds significance level with stars
             ci = TRUE)          # If TRUE, adds confidence intervals
```

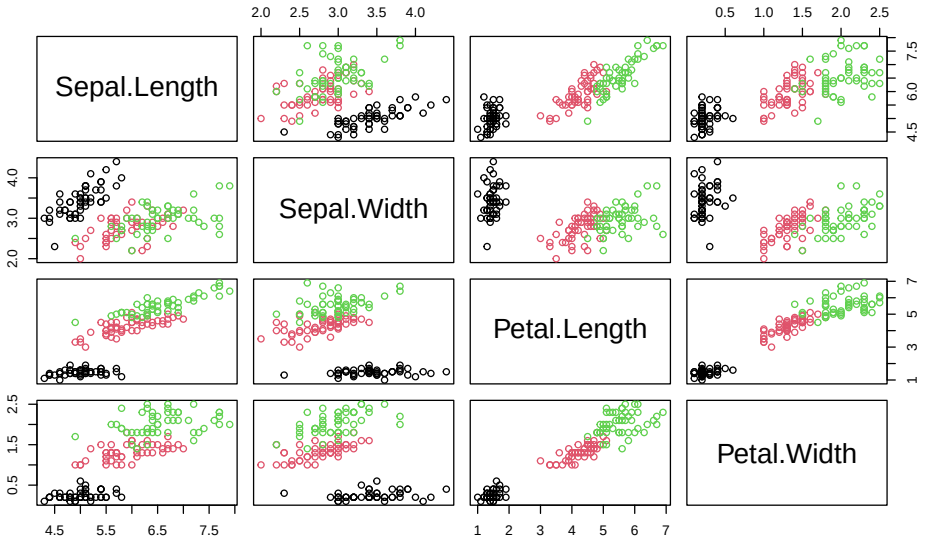



```
corPlot(data, cex = 1.2)
```




```
# Load the iris dataset
data(iris)

# Create a pairs plot using base R graphics
pairs(iris[, 1:4], col = as.numeric(iris$Species))
```



- How would you interpret the plot above? The first plot seems to be a scatter plot displaying the relationships between different features of the iris dataset. Unfortunately, the labels and legends are not clearly visible in the image. However, I can infer some general characteristics:

Data Distribution: The plot shows how the data points are distributed across different classes of the iris dataset. Feature Relationships: By examining the axes, we can deduce the relationships between different features, such as petal length, petal width, sepal length, and sepal width. Class Clustering: If the plot is color-coded by the iris species (Setosa, Versicolor, Virginica), we may observe how well the classes are separated based on the selected features.

The second plot appears to be a pairplot or scatterplot matrix displaying pairwise relationships between different features of the iris dataset. The following insights can be drawn:

Pairwise Relationships: Each scatterplot in the matrix represents the relationship between two features, allowing us to see how the variables interact with each other. Histograms: Along the diagonal, there may be histograms or kernel density estimates (KDEs) showing the univariate distribution of each feature. Class Separation: If the points are color-coded by species, the plots can show how well the species are differentiated based on the selected pairs of features. Correlation Patterns: By examining the scatterplots, one can identify potential correlations or linear relationships between different features.

Both plots provide a visual exploration of the iris dataset, aiding in understanding feature distributions, correlations, and class separations. The first plot offers a focused view of the relationships between specific features, while the second plot provides a comprehensive overview of all pairwise feature interactions.