01.Przygotowanie danych

W ramach warsztatu będziemy zajmować się modelowaniem spalania samochodów:

https://archive.ics.uci.edu/ml/datasets/Auto+MPG

Przydatne informacje:

- http://mlr-org.github.io/mlr-tutorial/devel/html/task/index.html
- http://mlr-org.github.io/mlr-tutorial/devel/html/preproc/index.html
- https://www.rdocumentation.org/packages/mlr/versions/2.10/topics/makeClassifTask

Wczytanie pliku CSV

W pliku mamy następujące kolumny:

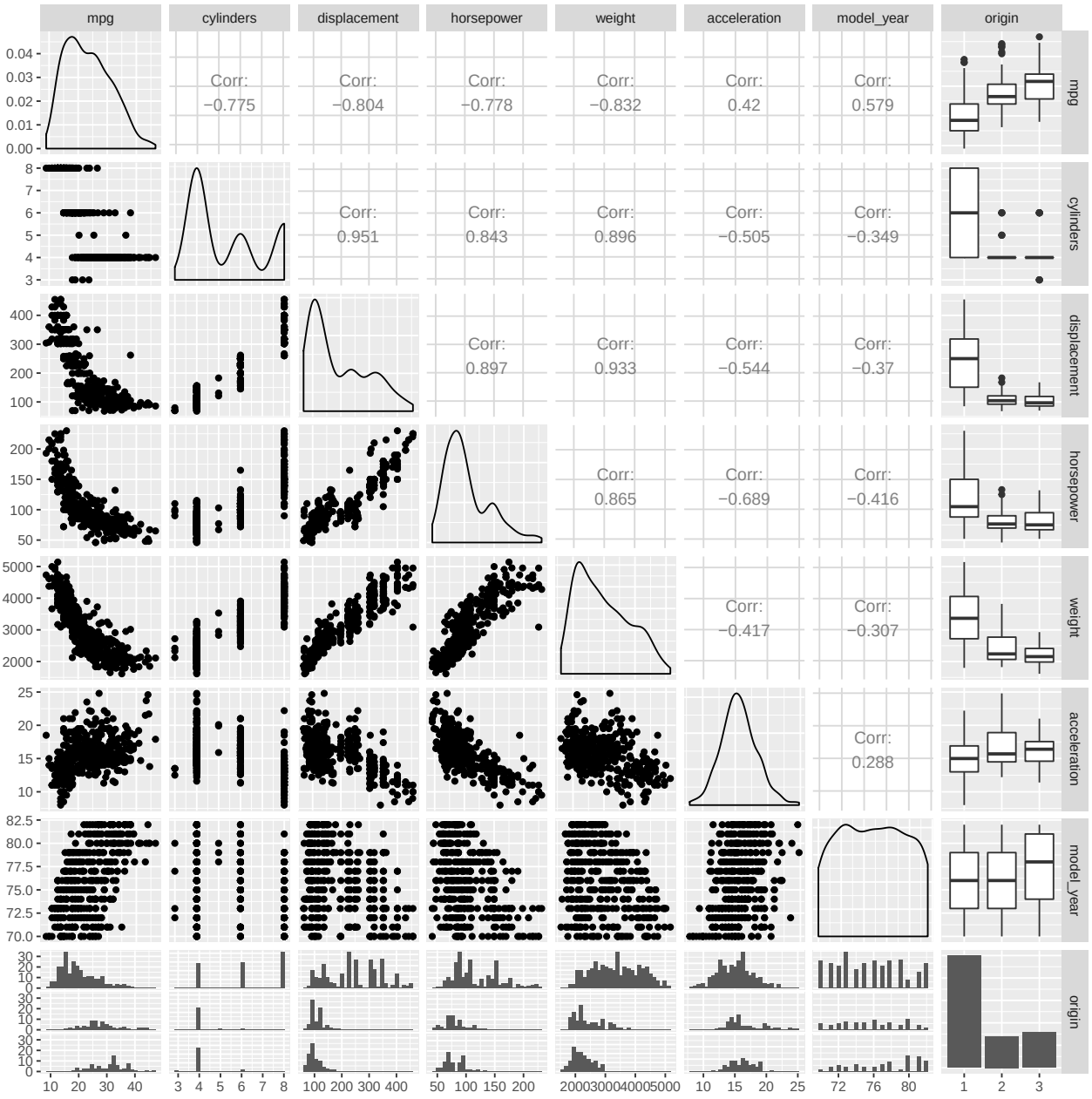
1. mpg: continuous (nasza zmienna celu)
2. cylinders: multi-valued discrete
3. displacement: continuous
4. horsepower: continuous
5. weight: continuous
6. acceleration: continuous
7. model year: multi-valued discrete
8. origin: multi-valued discrete
9. car name: string (identyfikator)

```
autoMpgDf <- readr::read_csv('data/auto-mpg.data.csv',na = '?') %>%
        dplyr::mutate(car_name = factor(car_name)
                      , origin = factor(origin))
summary(autoMpgDf)
```

```
##       mpg          cylinders      displacement     horsepower
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0
##  1st Qu.:17.50   1st Qu.:4.000   1st Qu.:104.2   1st Qu.: 75.0
##  Median :23.00   Median :4.000   Median :148.5   Median : 93.5
##  Mean   :23.51   Mean   :5.455   Mean   :193.4   Mean   :104.5
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:262.0   3rd Qu.:126.0
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0
##                                                  NA's   :6
##      weight      acceleration     model_year    origin
##  Min.   :1613   Min.   : 8.00   Min.   :70.00   1:249
##  1st Qu.:2224   1st Qu.:13.82   1st Qu.:73.00   2: 70
##  Median :2804   Median :15.50   Median :76.00   3: 79
##  Mean   :2970   Mean   :15.57   Mean   :76.01
##  3rd Qu.:3608   3rd Qu.:17.18   3rd Qu.:79.00
##  Max.   :5140   Max.   :24.80   Max.   :82.00
##
##            car_name
##  ford pinto    :  6
##  amc matador   :  5
##  ford maverick :  5
##  toyota corolla:  5
##  amc gremlin   :  4
##  amc hornet    :  4
```

```
##  (Other)        :369
```

```
GGally::ggpairs(autoMpgDf %>% dplyr::select(-car_name))
```



Utworzenie zadania regresji

Naszą zmienną celu jest `mpg`:

```
print(autoMpgTask)
```

```
## Supervised task: auto_mpg
## Type: regr
## Target: mpg
## Observations: 398
```

```
## Features:
## numerics  factors  ordered
##        6        2        0
## Missings: TRUE
## Has weights: FALSE
## Has blocking: FALSE
```

Przekształcanie atrybutów

Do wykonania mamy następujące kroki:

1. uzupełnienie brakujących danych,
2. wyrzucenie kolumny `car_name` (jest identyfikatorem wiersza),
3. zakodowanie wartości kolumny `origin` przy pomocy zmiennych wskaźnikowych ("one-hot-encoding"/"1-of-n"),
4. normalizacja (przez standaryzację) wartości parametrów.

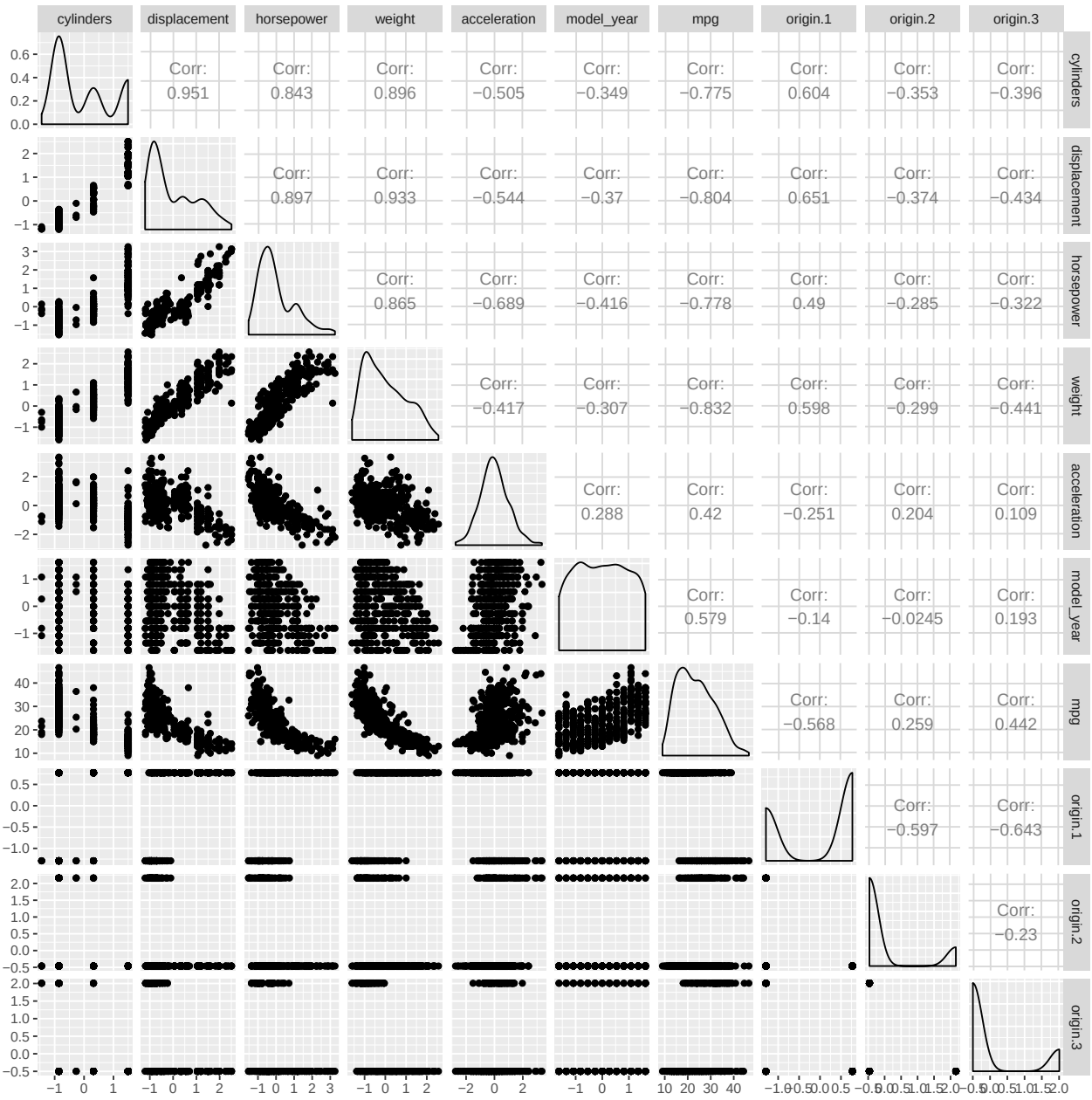
W efekcie otrzymamy zadanie:

```
autoMpgPreprocessedTask
```

```
## Supervised task: auto_mpg
## Type: regr
## Target: mpg
## Observations: 398
## Features:
## numerics  factors  ordered
##        9        0        0
## Missings: TRUE
## Has weights: FALSE
## Has blocking: FALSE
```

Wykres zadania po przeprocesowaniu:

```
GGally::ggpairs(autoMpgPreprocessedTask %>% getTaskData())
```



Zapisujemy nasze zadanie regresji na później:

```
#saveRDS(autoMpgPreprocessedTask,'data/01_task.RDS')
```

Selekcja atrybutów

Potrzebna nam jest metoda odpowiednia dla naszego zadania:

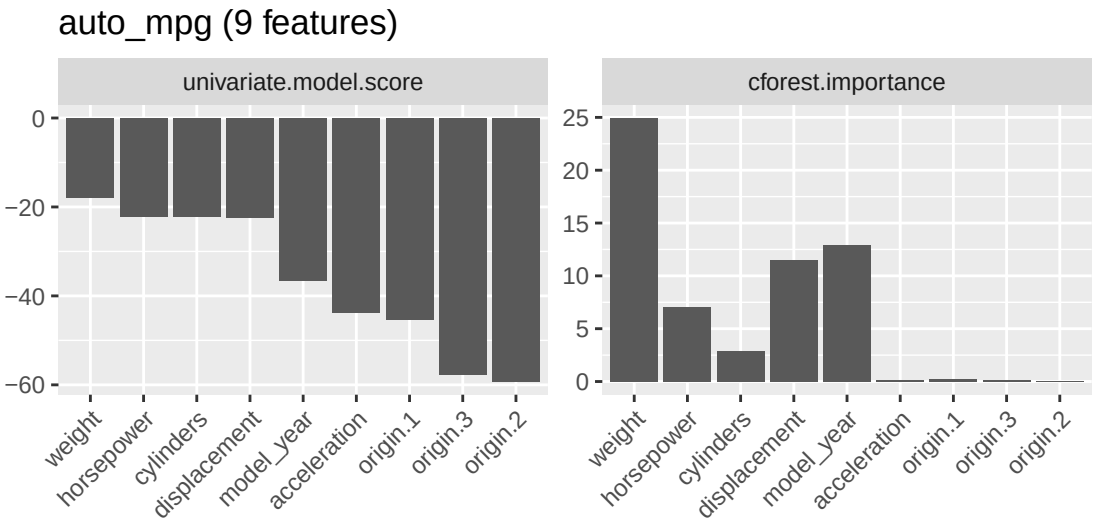
```
listFilterMethods(tasks=TRUE, features = TRUE) %>%
  dplyr::filter(task.regr==TRUE,  feature.ordered==TRUE, feature.numerics==TRUE) %>%
  dplyr::select(id, desc) %>% pander::pandoc.table()
```

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

```
##             id                          desc
## ------------------------- ------------------------------
##     cforest.importance       Permutation importance of
##                                random forest fitted in
##                                    package 'party'
##
##            mrmr              Minimum redundancy, maximum
##                                    relevance filter
##
##   permutation.importance   Aggregated difference between
##                                  feature permuted and
##                                 unpermuted predictions
##
##   randomForestSRC.rfsrc    Importance of random forests
##                                    fitted in package
##                            'randomForestSRC'. Importance
##                            is calculated using argument
##                                     'permute'.
##
## randomForestSRC.var.select  Minimal depth of / variable
##                            hunting via method var.select
##                             on random forests fitted in
##                             package 'randomForestSRC'.
##
##   univariate.model.score   Resamples an mlr learner for
##                                 each input feature
##                            individually. The resampling
##                            performance is used as filter
##                            score, with rpart as default
##                                      learner.
## ------------------------------------------------------
```

Przygotujmy teraz wykresy istotności cech dla kilku miar:

```
plotFilterValues(featureImportance)
```



Wyniki selekcji można też obejrzeć bardziej interaktywnie:

```
plotFilterValuesGGVIS(featureImportance)
```

Dodatkowe ćwiczenia:

1. czy w naszym zbiorze danych są elementy odstające? jak można sobie radzić w takich przypadkach korzystając z MLR?
2. w jaki sposób można “wyrzucić” atrybuty z zadania?