Own Project - Predicting Peptide Retention Times

1. Introduction/overview/executive summary

Chromatography is used in proteomics experiments to slowly elute different peptide species for the subsequent mass spectrometric analysis. Goal of the here described study is to predict the retention time (RT) of peptides based only on their amino acid sequence. The data set containing peptide sequences and RT was downloaded from:

https://www.kaggle.com/kirillpe/proteomics-retention-time-prediction (https://www.kaggle.com/kirillpe/proteomics-retention-time-prediction)

The data is also included in the uploaded files (unmod.txt).

General information about proteomics:

https://en.wikipedia.org/wiki/Proteomics (https://en.wikipedia.org/wiki/Proteomics)

2. Methods/analysis

The provided peptide sequences were derived by digestion of proteins and the amino acid sequences of the peptides can be used to predict the physico-chemical properties of the peptides. General information about amino acids can be found here:

https://en.wikipedia.org/wiki/Amino_acid (https://en.wikipedia.org/wiki/Amino_acid)

The R package “Peptides” is used to predict physico-chemical properties of all peptides.

Manual of the Peptides R package:

https://cran.r-project.org/web/packages/Peptides/Peptides.pdf (https://cran.r-project.org/web/packages/Peptides/Peptides.pdf)

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-pr
oject.org")
```

```
## Loading required package: tidyverse
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ─
─
```

```
## ✓ ggplot2 3.3.5     ✓ purrr   0.3.4
## ✓ tibble  3.1.3     ✓ dplyr   1.0.7
## ✓ tidyr   1.1.3     ✓ stringr 1.4.0
## ✓ readr   2.0.0     ✓ forcats 0.5.1
```

```
## ── Conflicts ─────────────────────────────────────────── tidyverse_conflicts() ─
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: data.table
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
if(!require(Peptides)) install.packages("Peptides", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: Peptides
```

```
if(!require(ggpubr)) install.packages("ggpubr", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: ggpubr
```

```
if(!require(broom)) install.packages("broom", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: broom
```

```
if(!require(glue)) install.packages("glue", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: glue
```

```
##
## Attaching package: 'glue'
```

```
## The following object is masked from 'package:dplyr':
##
##     collapse
```

```
library(tidyverse)
library(caret)
library(data.table)
library(Peptides)
library(ggpubr)
library(broom)
library(glue)

# Load data and get a first overview
peptide_rt <- read_delim("unmod.txt", delim = '\t', show_col_types = FALSE)
head(peptide_rt)
```

```
## # A tibble: 6 × 2
##   sequence                          RT
##   <chr>                          <dbl>
## 1 VSLDDLQQSIEEDEDHVQSTDIAAMQK    15083.
## 2 NVIAETGAGQHGVATATACAK           6098.
## 3 FATVPTGGASSAAAGAAGAAAGGDAAEEEK  8934.
## 4 LTPAANQVEIHPLLPQDELINFCK       15086.
## 5 DAGAISGLNVLRIINEPTAAAIAYGLGAGK 15405
## 6 FALAGAIGCGSTHSSMVPIDVVK        13899.
```

```
str(peptide_rt)
```

```
## spec_tbl_df [14,361 × 2] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ sequence: chr [1:14361] "VSLDDLQQSIEEDEDHVQSTDIAAMQK" "NVIAETGAGQHGVATATACAK
" "FATVPTGGASSAAAGAAGAAAGGDAAEEEK" "LTPAANQVEIHPLLPQDELINFCK" ...
##  $ RT      : num [1:14361] 15083 6098 8934 15086 15405 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   sequence = col_character(),
##   ..   RT = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Test of some functions from the Peptides package on the 20 natural amino acids.

```
# Calculate the charge of amino acids at pH 2.7 (default pH for peptide chromatogr
aphy)
sort(c(sapply(aaList(), function(x) charge(x, pH=2.7))))
```

```
##         D         E         C         Y         A         F         G         I
## 0.2029885 0.2764599 0.3038678 0.3038711 0.3038711 0.3038711 0.3038711 0.3038711
##         L         M         N         P         Q         S         T         V
## 0.3038711 0.3038711 0.3038711 0.3038711 0.3038711 0.3038711 0.3038711 0.3038711
##         W         H         K         R
## 0.3038711 1.3033702 1.3038711 1.3038711
```

```
# Calculate hydrophobicity for amino acids
sort(c(sapply(aaList(), hydrophobicity)))
```

```
##    R    K    D    E    N    Q    H    P    Y    W    S    T    G    A    M    C
## -4.5 -3.9 -3.5 -3.5 -3.5 -3.5 -3.2 -1.6 -1.3 -0.9 -0.8 -0.7 -0.4  1.8  1.9  2.5
##    F    L    V    I
##  2.8  3.8  4.2  4.5
```

```
# Calculate molecular weight
sort(c(sapply(aaList(), mw)))
```

```
##         G         A         S         P         V         T         C         I
##  75.06714  89.09404 105.09344 115.13194 117.14784 119.12034 121.15404 131.17464
##         L         N         D         Q         K         E         M         H
## 131.17464 132.11904 133.10384 146.14594 146.18934 147.13074 149.20784 155.15634
##         F         R         Y         W
## 165.19184 174.20274 181.19124 204.22844
```

```
# Composition of amino acids in data set
aaComp(paste(peptide_rt$sequence, collapse = ""))
```

```
## [[1]]
##           Number  Mole%
## Tiny       51331 27.286
## Small      96821 51.467
## Aliphatic  53365 28.367
## Aromatic   17373  9.235
## NonPolar   91995 48.901
## Polar      96129 51.099
## Charged    51441 27.344
## Basic      22292 11.850
## Acidic     29149 15.495
```

Based on the peptide sequence, different characteristics of the peptides are calculated and predicted.

```
# Calculating and predicting peptide features
peptide_rt <- peptide_rt %>%
  mutate(length = str_length(sequence), # adding lenght of peptide
         charge = charge(sequence, pH=2.7), # adding charge
         hydrophobicity = hydrophobicity(sequence), # adding hydrophobicity
         molecular_weight = mw(sequence), # adding molecular weight
         aliphatic_index = aIndex(sequence), #adding aliphatic index
         )
head(peptide_rt)
```

```
## # A tibble: 6 × 7
##   sequence      RT length charge hydrophobicity molecular_weight aliphatic_inde
x
##   <chr>      <dbl>  <int>  <dbl>          <dbl>            <dbl>           <dbl
>
## 1 VSLDDLQQ… 15083.     27   1.72       -0.804              3045.            86.
7
## 2 NVIAETGA…  6098.     21   2.28        0.252              1970.            74.
8
## 3 FATVPTGG…  8934.     30   1.12        0.123              2564.            49.
7
## 4 LTPAANQV… 15086.     24   2.15        0.00833            2704.           118.
## 5 DAGAISGL… 15405      30   2.18        0.58               2897.           124
## 6 FALAGAIG… 13899.     23   2.20        0.935              2261.           102.
```

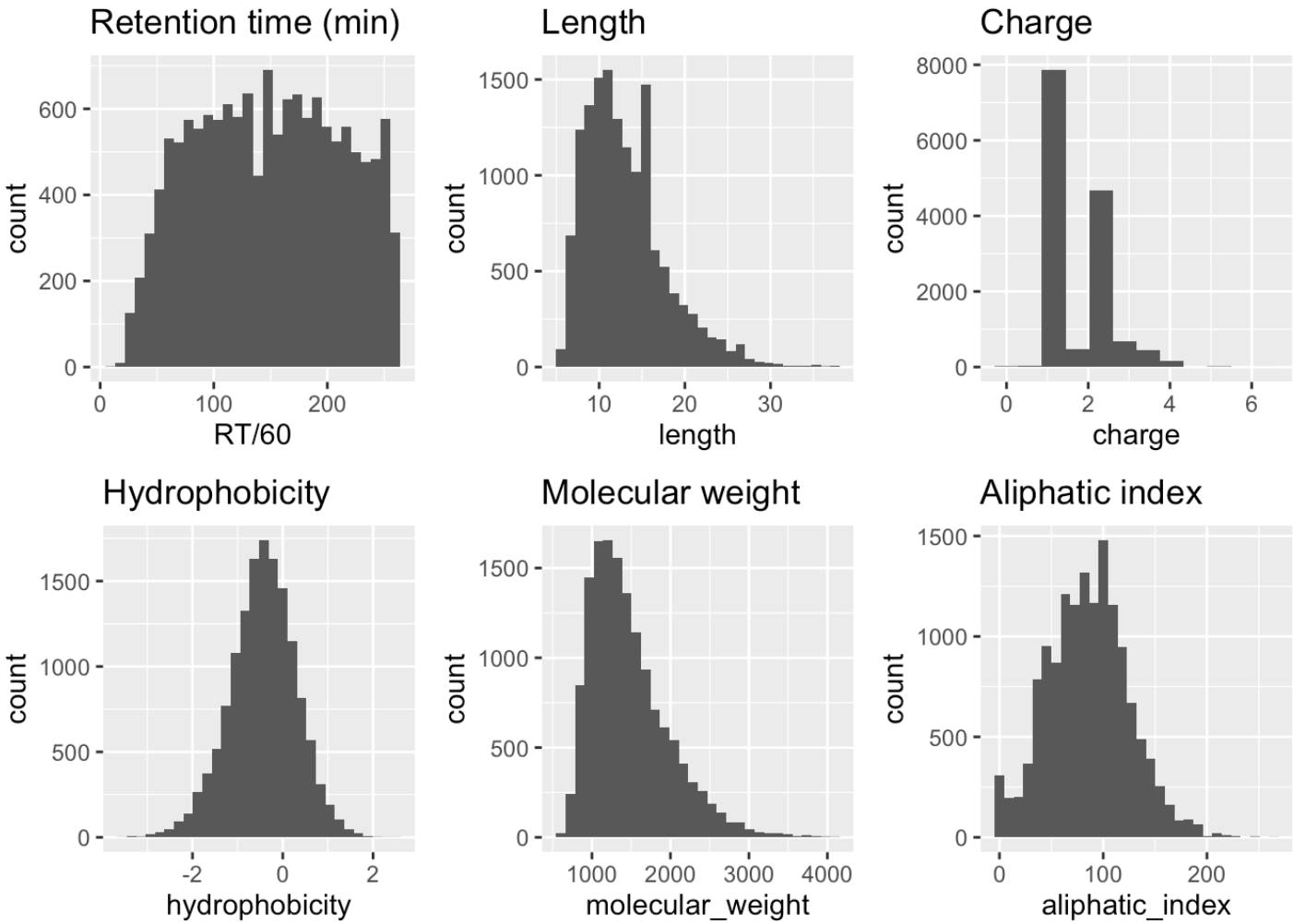
Calculate the median peptide length

```
median(peptide_rt$length)
```

```
## [1] 12
```

The RT of peptides in minutes as well as the calculated/predicted peptide features are plotted as histograms.

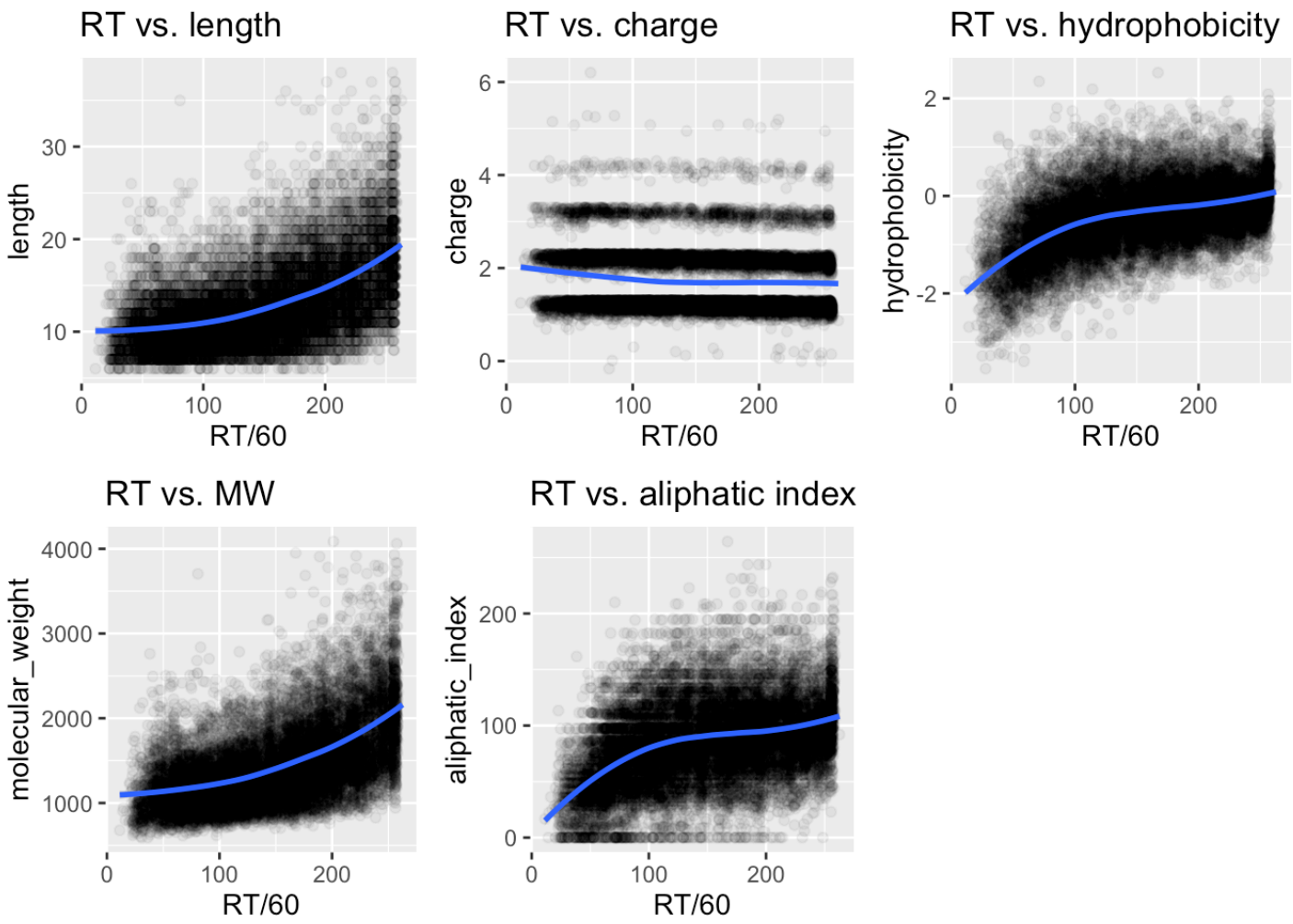
```
pep_rt <- peptide_rt %>%
  ggplot(aes(x = RT/60)) +
  geom_histogram(bins = 30) +
  ggtitle("Retention time (min)")

pep_length <- peptide_rt %>%
  ggplot(aes(x = length)) +
  geom_histogram(bins = 30) +
  ggtitle("Length")

pep_charge <- peptide_rt %>%
  ggplot(aes(x = charge)) +
  geom_histogram(bins = 12) +
  ggtitle("Charge")

pep_hydrophobicity <- peptide_rt %>%
  ggplot(aes(x = hydrophobicity)) +
  geom_histogram(bins = 30) +
  ggtitle("Hydrophobicity")

pep_mw <- peptide_rt %>%
  ggplot(aes(x = molecular_weight)) +
  geom_histogram(bins = 30) +
  ggtitle("Molecular weight")

pep_aliphatic <- peptide_rt %>%
  ggplot(aes(x = aliphatic_index)) +
  geom_histogram(bins = 30) +
  ggtitle("Aliphatic index")

ggarrange(pep_rt, pep_length, pep_charge,
          pep_hydrophobicity, pep_mw, pep_aliphatic,
          ncol = 3, nrow = 2)
```



Scatter plots are used to show the relationships of RT to the different peptide features.

```
pep_rt_length <- peptide_rt %>%
  ggplot(aes(x = RT/60, y = length)) +
  geom_point(alpha = 0.05) +
  ggtitle("RT vs. length") +
  geom_smooth(method = "loess")

pep_rt_charge <- peptide_rt %>%
  ggplot(aes(x = RT/60, y = charge)) +
  geom_point(alpha = 0.05) +
  ggtitle("RT vs. charge") +
  geom_smooth(method = "loess")

pep_rt_hydrophobicity <- peptide_rt %>%
  ggplot(aes(x = RT/60, y = hydrophobicity)) +
  geom_point(alpha = 0.05) +
  ggtitle("RT vs. hydrophobicity") +
  geom_smooth(method = "loess")

pep_rt_mw <- peptide_rt %>%
  ggplot(aes(x = RT/60, y = molecular_weight)) +
  geom_point(alpha = 0.05) +
  ggtitle("RT vs. MW") +
  geom_smooth(method = "loess")

pep_rt_aliphatic <- peptide_rt %>%
  ggplot(aes(x = RT/60, y = aliphatic_index)) +
  geom_point(alpha = 0.05) +
  ggtitle("RT vs. aliphatic index") +
  geom_smooth(method = "loess")

ggarrange(pep_rt_length, pep_rt_charge, pep_rt_hydrophobicity,
          pep_rt_mw, pep_rt_aliphatic,
          ncol = 3, nrow = 2)
```

```
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
```




```
# Split data in train and test set. Test set is 10% of peptide_rt data.
suppressWarnings(set.seed(1, sample.kind="Rounding"))
test_index <- createDataPartition(y = peptide_rt$RT, times = 1, p = 0.1, list = FA
LSE)
train_data <- peptide_rt[-test_index,]
dim(train_data)
```

```
## [1] 12923     7
```

```
test_data <- peptide_rt[test_index,]
dim(test_data)
```

```
## [1] 1438    7
```

Calculate the Root Mean Square Error (RMSE) of RT for linear models with different numbers of peptide features.

```
# Calculate the RMSE for predicting RT with peptide length
fit_l <- lm(RT ~ length, data = train_data)
RMSE(predict(fit_l, test_data), test_data$RT)/60
```

```
## [1] 53.00806
```

```
# Calculate the RMSE for predicting RT with peptide hydrophobicity
fit_h <- lm(RT ~ hydrophobicity, data = train_data)
RMSE(predict(fit_h, test_data), test_data$RT)/60
```

```
## [1] 55.60513
```

```
# Calculate the RMSE for predicting RT with peptide molecular weight
fit_m <- lm(RT ~ molecular_weight, data = train_data)
RMSE(predict(fit_m, test_data), test_data$RT)/60
```

```
## [1] 51.66473
```

```
# Calculate the RMSE for predicting RT with peptide aliphatic index
fit_a <- lm(RT ~ aliphatic_index, data = train_data)
RMSE(predict(fit_a, test_data), test_data$RT)/60
```

```
## [1] 58.45291
```

```
# Calculate the RMSE for predicting RT with peptide charge
fit_c <- lm(RT ~ charge, data = train_data)
RMSE(predict(fit_c, test_data), test_data$RT)/60
```

```
## [1] 62.87771
```

```
# Calculate the RMSE for predicting RT with all available peptide features
fit_lhmac <- lm(RT ~ length + hydrophobicity + molecular_weight + aliphatic_index
+ charge, data = train_data)
RMSE(predict(fit_lhmac, test_data), test_data$RT)/60
```

```
## [1] 29.75443
```

The linear model gave the smallest RMSE when all peptide features were included. Thus, all peptide features will be used for testing additional models.

3. Results

The following models are evaluated to predict the RT of peptides in order to find the model that can best predict the RT with the smallest RMSE for the test data set:

- glm: Generalized Linear Model
- knn: k-Nearest Neighbour Classification
- svmLinear: linear support vector machines
- rf: random forest

In addition to calculating the RMSE for the test set, the time to train the model will also be taken.

```
# Define models to be tested and create empty vectors to record data
models <- c("glm", "knn", "svmLinear", "rf")
model_rmse <- numeric()
sec_per_model <- numeric()

# Train all models, take the time for each training and calculate RMSE on
# predictions for the test set
for (model in models){
  tic <- Sys.time()
  suppressWarnings(fit <- train(RT ~ length + hydrophobicity +
    molecular_weight + aliphatic_index + charge,
      method = model,
      data = train_data))
  toc <- Sys.time()
    duration <- round(as.numeric(difftime(toc, tic, units="secs")),1)
    sec_per_model[model] <- duration

  rmse <- RMSE(predict(fit, test_data), test_data$RT)/60
  model_rmse[model] <- rmse
}

# Overview of time needed to train each model in seconds
sec_per_model
```

```
##       glm       knn svmLinear        rf
##       2.0      19.3    3824.4   20056.0
```

```
# Overview RMSE in min for RT predictions for the test set for each model
model_rmse
```

```
##       glm       knn svmLinear        rf
##  29.75443  44.01036  29.88969  27.54048
```

4. Conclusion

Four different models (glm: Generalized Linear Model, knn: k-Nearest Neighbour Classification, svmLinear: linear support vector machines, rf: random forest) were tested to predict the retention time of peptides based only on the amino acid sequence and predicted/calculated physico-chemical properties of the peptides. The rf model achieved the best results with the lowest RMSE of all four models. However, the time to train the rf model was significantly longer (more than one hour) compared to other models (only seconds). As seen in the exploratory data analysis, there is a nearly linear relationship between the RT and some peptide features. Therefore, it is no surprise that also the glm model gave a good result (second smallest RMSE) with a very short time for training the model.

Andreas Hofmann
29-Dec-2021

———