Machine learning EDA log

Dennis Haandrikman

27/09/2022

Contents

Introduction **2**

Exploring the background of the data 2

Importing the data 3

EDA exploration 7

Codebook 7

Univariate analysis 8

Multivariate analysis 17

Export data 21

Introduction

In modern day science, machine learning is playing a more crucial step by the year, allowing predictions of topics which would take years to discover with the help of experiments. However, a caveat to this is, that the prediction is only as good as the submitted data and the programmer who coded the algorythm for machine learning.

As such it is important for one to understand what one can do with the data they've received and that they know the background of the data itself. As such it is important to garner background information about the data and to explore what's possible with the received data by the help of an EDA.

Exploring the background of the data

It would be important to explore what the meaning and use for each data variable is before we start working with the data.

First off we'll discuss what the origin of the data files is, the paper that this dataset was used in, references IEDB(http://tools.iedb.org/bcell/help/) as the main source for the peptide sequences, due to the website containing information about if an amino acid peptide exhibited anti-body inducing activty, which was marked as the target label. So both the bcel & sars peptide data have been sourced from IEDB, with the protein properties being filled in with help of the Uniprot website. The covid peptide/protein data has been calculated with help of the same websites, except for that these haven't been labelled by IEDB.

So in the data-files the following columns are available and will be explained below:

1. parent_protein_id; This is the parent protein ID, with which it can be found on Uniprot. It's a string containing a protein ID.
2. protein_seq; This is the protein sequence of the full protein the peptide belongs to. It's a string that consists of amino acids.
3. start_position; This is the starting position of the peptide in correspondence to the full protein sequence given in protein_seq. It's an integer.
4. end_position; This is the ending position of the peptide in correspondence to the full protein sequence given in protein_seq. It's an integer.
5. peptide_seq; This is the peptide sequence that is to be analyzed. It's a string containing amino acids.
6. chou_fasman; This is a calculated peptide feature by the IEDB website, it predicts if there is a ß turn in the peptide. It's a double.
7. emini; This is a calculated peptide feature by the IEDB website, it calculates how much relative surface accessibility is available on the peptide. It's a double
8. kolaskar_tongaonkar; This is a calculated peptide prediction by the IEDB website in how antigenic the peptide is. It's a double.
9. parker; This is a calculated peptide feature by the IEDB website in how hydrophobic the peptide is. It's a double.
10. isoelectric_point; This is the calculated pH at which the charge of the protein is zero. It's a double.
11. aromacity; This is a calculated protein feature in how many aromatic connections there are in the protein. It's a double.
12. hydrophobicity; This is a calculated protein feature in how hydrophobic the complete protein is. It's a double.
13. stability; This is a calculated protein feature that refers to the energy difference between the folded and unfolded state of the protein. It's a double.

Importing the data

15/09/2022, 16/09/2022, 20/09/2022 After acquiring the data from the database, import the data to see if there were any inconsistencies.

```
# Make sure the working directory is the main folder containing all the folders (project folder).
# Get the data files and import them.
bcel_data <- tibble(read.csv("../Data/input_bcell.csv", header = TRUE))
sars_data <- tibble(read.csv("../Data/input_sars.csv", header = TRUE))
covid_data <- tibble(read.csv("../Data/input_covid.csv", header = TRUE))

# print a quick head for the tables to see if it imported properly.
# Due to protein seq being such a lengthy tab, excluding from the head to make the head show able.
# In each dataset, the protein seq is the second column.
pander(head(bcel_data[-2]))
```

Table 1: Table continues below

parent_protein_id	start_position	end_position	peptide_seq	chou_fasman
A2T3T0	161	165	SASFT	1.016
F0V2I4	251	255	LCLKI	0.77
O75508	145	149	AHRET	0.852
O84462	152	156	SNYDD	1.41
P00918	85	89	DGTYR	1.214
P00918	155	159	GLQKV	0.928

Table 2: Table continues below

emini	kolaskar_tongaonkar	parker	isoelectric_point	aromaticity
0.703	1.018	2.22	5.81	0.1033
0.179	1.199	-3.86	6.211	0.06548
3.427	0.96	4.28	8.224	0.09179
2.548	0.936	6.32	4.238	0.04478
1.908	0.937	4.64	6.867	0.1038
0.547	1.09	0.9	6.867	0.1038

hydrophobicity	stability	target
-0.1438	40.27	1
-0.0369	25	1
0.8792	27.86	1
-0.5214	30.77	1
-0.5788	21.68	1
-0.5788	21.68	1

```
pander(head(sars_data[-2]))
```

Table 4: Table continues below

parent_protein_id	start_position	end_position	peptide_seq
AAU93319	1	17	MFIFLLFLTLTSGSDLD
AAU93319	1	15	MFIFLLFLTLTSGSD
AAU93319	2	10	FIFLLFLTL
AAU93319	6	20	LFLTLTSGSDLDRCT
AAU93319	9	25	TLTSGSDLDRCTTFDDV
AAU93319	11	25	TSGSDLDRCTTFDDV

Table 5: Table continues below

chou_fasman	emini	kolaskar_tongaonkar	parker	isoelectric_point
0.887	0.04	1.056	-2.159	5.57
0.869	0.047	1.056	-2.5	5.57
0.621	0.042	1.148	-7.467	5.57
1.021	0.23	1.049	0.927	5.57
1.089	0.627	1.015	3.165	5.57
1.131	0.848	1.007	3.853	5.57

aromaticity	hydrophobicity	stability	target
0.1163	-0.06112	33.21	0
0.1163	-0.06112	33.21	0
0.1163	-0.06112	33.21	0
0.1163	-0.06112	33.21	0
0.1163	-0.06112	33.21	0
0.1163	-0.06112	33.21	0

```
pander(head(covid_data[-2]))
```

Table 7: Table continues below

parent_protein_id	start_position	end_position	peptide_seq	chou_fasman
6VYB_A	1	5	MGILP	0.948
6VYB_A	2	6	GILPS	1.114
6VYB_A	3	7	ILPSP	1.106
6VYB_A	4	8	LPSPG	1.324
6VYB_A	5	9	PSPGM	1.326
6VYB_A	6	10	SPGMP	1.326

Table 8: Table continues below

emini	kolaskar_tongaonkar	parker	isoelectric_point	aromaticity
0.28	1.033	-2.72	6.036	0.1093
0.379	1.07	-0.58	6.036	0.1093
0.592	1.108	-1.3	6.036	0.1093

emini	kolaskar_tongaonkar	parker	isoelectric_point	aromaticity
0.836	1.053	1.44	6.036	0.1093
1.004	0.968	2.44	6.036	0.1093
1.004	0.968	2.44	6.036	0.1093

hydrophobicity	stability
-0.1386	31.38
-0.1386	31.38
-0.1386	31.38
-0.1386	31.38
-0.1386	31.38
-0.1386	31.38

```
# here's the max length of a protein sequence to showcase why it was excluded from the head
max(nchar(bcel_data[2,]))
```

```
## [1] 336
```

As we can see the information has been imported correctly. However, it is of importance to check for any NA values in the information. As such, an is.na() check will be performed on the data

```
# Using any(is.na(x)) showcases in true/false if there are any NA values.
any(is.na(bcel_data))
```

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

```
any(is.na(sars_data))
```

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

```
any(is.na(covid_data))
```

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

As we can see there aren't any missing values whatsoever in the datasets, as such we can continue to the actual EDA after we check the value-types each column has

```
# Information for the bcel_data dataset
str(bcel_data)
```

```
## tibble [14,387 x 14] (S3: tbl_df/tbl/data.frame)
##  $ parent_protein_id  : chr [1:14387] "A2T3T0" "F0V2I4" "O75508" "O84462" ...
##  $ protein_seq        : chr [1:14387] "MDVLYSLSKTLKDARDKIVEGTLYSNVSDLIQQFNQMIITMNGNEFQTGGIGNLPIRNWNF
##  $ start_position     : int [1:14387] 161 251 145 152 85 155 22 253 218 261 ...
##  $ end_position       : int [1:14387] 165 255 149 156 89 159 26 257 222 265 ...
##  $ peptide_seq        : chr [1:14387] "SASFT" "LCLKI" "AHRET" "SNYDD" ...
##  $ chou_fasman        : num [1:14387] 1.016 0.77 0.852 1.41 1.214 ...
```

```
##  $ emini              : num [1:14387] 0.703 0.179 3.427 2.548 1.908 ...
##  $ kolaskar_tongaonkar: num [1:14387] 1.018 1.199 0.96 0.936 0.937 ...
##  $ parker             : num [1:14387] 2.22 -3.86 4.28 6.32 4.64 0.9 2.66 2 4.62 0.42 ...
##  $ isoelectric_point  : num [1:14387] 5.81 6.21 8.22 4.24 6.87 ...
##  $ aromaticity        : num [1:14387] 0.1033 0.0655 0.0918 0.0448 0.1038 ...
##  $ hydrophobicity     : num [1:14387] -0.1438 -0.0369 0.8792 -0.5214 -0.5788 ...
##  $ stability          : num [1:14387] 40.3 25 27.9 30.8 21.7 ...
##  $ target             : int [1:14387] 1 1 1 1 1 1 1 1 1 1 ...
```

```
# Information for the sars_data dataset
str(sars_data)
```

```
## tibble [520 x 14] (S3: tbl_df/tbl/data.frame)
##  $ parent_protein_id  : chr [1:520] "AAU93319" "AAU93319" "AAU93319" "AAU93319" ...
##  $ protein_seq        : chr [1:520] "MFIFLLFLTLTSGSDLDRCTTFDDVQAPNYTQHTSSMRGVYYPDEIFRSDTLYLTQDLFLPFYS
##  $ start_position     : int [1:520] 1 1 2 6 9 11 16 17 19 19 ...
##  $ end_position       : int [1:520] 17 15 10 20 25 25 30 33 48 38 ...
##  $ peptide_seq        : chr [1:520] "MFIFLLFLTLTSGSDLD" "MFIFLLFLTLTSGSD" "FIFLLFLTL" "LFLTLTSGSDLDR
##  $ chou_fasman        : num [1:520] 0.887 0.869 0.621 1.021 1.089 ...
##  $ emini              : num [1:520] 0.04 0.047 0.042 0.23 0.627 ...
##  $ kolaskar_tongaonkar: num [1:520] 1.06 1.06 1.15 1.05 1.01 ...
##  $ parker             : num [1:520] -2.159 -2.5 -7.467 0.927 3.165 ...
##  $ isoelectric_point  : num [1:520] 5.57 5.57 5.57 5.57 5.57 ...
##  $ aromaticity        : num [1:520] 0.116 0.116 0.116 0.116 0.116 ...
##  $ hydrophobicity     : num [1:520] -0.0611 -0.0611 -0.0611 -0.0611 -0.0611 ...
##  $ stability          : num [1:520] 33.2 33.2 33.2 33.2 33.2 ...
##  $ target             : int [1:520] 0 0 0 0 0 0 0 1 0 1 ...
```

```
#Information for the covid_data dataset
str(covid_data)
```

```
## tibble [20,312 x 13] (S3: tbl_df/tbl/data.frame)
##  $ parent_protein_id  : chr [1:20312] "6VYB_A" "6VYB_A" "6VYB_A" "6VYB_A" ...
##  $ protein_seq        : chr [1:20312] "MGILPSPGMPALLSLVSLLSVLLMGCVAETGTQCVNLTTRTQLPPAYTNSFTRGVYYPDKV
##  $ start_position     : int [1:20312] 1 2 3 4 5 6 7 8 9 10 ...
##  $ end_position       : int [1:20312] 5 6 7 8 9 10 11 12 13 14 ...
##  $ peptide_seq        : chr [1:20312] "MGILP" "GILPS" "ILPSP" "LPSPG" ...
##  $ chou_fasman        : num [1:20312] 0.948 1.114 1.106 1.324 1.326 ...
##  $ emini              : num [1:20312] 0.28 0.379 0.592 0.836 1.004 ...
##  $ kolaskar_tongaonkar: num [1:20312] 1.033 1.07 1.108 1.053 0.968 ...
##  $ parker             : num [1:20312] -2.72 -0.58 -1.3 1.44 2.44 2.44 1.56 -0.7 -3.68 -1.54 ...
##  $ isoelectric_point  : num [1:20312] 6.04 6.04 6.04 6.04 6.04 ...
##  $ aromaticity        : num [1:20312] 0.109 0.109 0.109 0.109 0.109 ...
##  $ hydrophobicity     : num [1:20312] -0.139 -0.139 -0.139 -0.139 -0.139 ...
##  $ stability          : num [1:20312] 31.4 31.4 31.4 31.4 31.4 ...
```

We can see that the covid_data dataset doesn't contain the target column at the end of the dataset, this is to be expected as the original use of the dataset was to calculate the target values for the covid_data. Next to that we can see that all the information has been loaded in properly.

However, due to the b-cell & sars datasets were both used for the same use; training the model and contain the exact same columns, it would be wise to merge the 2 datasets in 1 so that it's easier to manage the data from the 2 sets.

```r
#combining the 2 datasets with use of rbind
bcel_sars <- rbind(bcel_data,sars_data)
# To see if the binding went properly, we're calling going to calculate the length of the dataframes be
# Calculating the supposed combined length of the new dataset
print(paste("The number of columns we're expecting is: ", nrow(bcel_data) + nrow(sars_data)))
```

```
## [1] "The number of columns we're expecting is:  14907"
```

```r
print(paste("The number of columns that the new dataset contains: ", nrow(bcel_sars)))
```

```
## [1] "The number of columns that the new dataset contains:  14907"
```

As we can see above is that combining the 2 datasets was successful and no columns have been lost.

At the very end of the data import and clean-up, it would be wise to convert target column in the dataset from an integer to a factor. This is so we can differentiate between the non-antibody promoting sequences and antibody promoting sequences.

```r
bcel_sars$target <- factor(bcel_sars$target, labels = c("Non-promoting", "Promoting"))

pander(data.frame(Before=bcel_data[5:15,14], After=bcel_sars[5:15,14]))
```

target	target.1
1	Promoting
1	Promoting
1	Promoting
1	Promoting
1	Promoting
1	Promoting
1	Promoting
1	Promoting
0	Non-promoting
0	Non-promoting
0	Non-promoting

As we can see from the quick splice in the dataframe above, the value of 0 has been converted to Non-promoting and the value of 1 has been converted to Promoting. This has been completed as expected.

EDA exploration

20/09/2022, 23/09/2022, 25/09/2022, 26/09/2022, 01/10/2022, 04/10/2022, 05/10/2022

Codebook

Now that we've made sure all the datasets have been imported properly and there are no missing values in the datasets, it was time to perform an EDA exploration.

Since the datasets all go over the properties of the peptide/proteins, a codebook was set up with information on all the properties.

```
code_book <- tibble(read.csv("../peptide_protein_properties_codebook.csv", header=TRUE,sep=","))

pander(head(code_book))
```

Table 11: Table continues below

Name	Full.name	Label
parent_protein_id	Parent protein ID	Parent protein ID
protein_seq	Parent protein sequence	Parent protein sequence
start_position	Start position of peptide	Start position of peptide
end_position	End position of peptide	End position of peptide
peptide_seq	Peptide sequence	Peptide sequence
chou_fasman	Chou-Fasman method	Chou-Fasman scale

Data.Type	Unit	Description
character	NA	The ID for the protein
character	NA	The amino acid sequence for the protein
numeric	NA	The starting position of the peptide
numeric	NA	The ending position of the peptide
character	NA	The amino acid sequence for the peptide fragment
numeric	NA	A scale for predicting turns to predict antibody epitopes

As we can see the code-book has been imported properly, the NA's in unit have been input correctly as there is no unit for those values.

Univariate analysis

A summary for the data would be nice to oversee, however, not all the columns would be crucial to undergo a summary, for example the sequence ID, peptide ID, sequence can be omitted due to not being measurable data. As such the summary has only been performed on the protein characteristics.

```
#Summary of the Bcel-data for the peptide/protein properties
pander(summary(bcel_sars[6:13]))
```

Table 13: Table continues below

chou_fasman	emini	kolaskar_tongaonkar	parker
Min. :0.5340	Min. : 0.000	Min. :0.838	Min. :-9.029
1st Qu.:0.9130	1st Qu.: 0.244	1st Qu.:0.987	1st Qu.: 0.600
Median :0.9910	Median : 0.551	Median :1.021	Median : 1.775
Mean :0.9949	Mean : 1.083	Mean :1.022	Mean : 1.750
3rd Qu.:1.0730	3rd Qu.: 1.208	3rd Qu.:1.055	3rd Qu.: 2.960
Max. :1.5460	Max. :40.605	Max. :1.255	Max. : 9.120

isoelectric_point	aromaticity	hydrophobicity	stability
Min. : 3.686	Min. :0.00000	Min. :-1.9712	Min. : 5.449
1st Qu.: 5.570	1st Qu.:0.06250	1st Qu.:-0.6001	1st Qu.: 31.726
Median : 6.448	Median :0.07595	Median :-0.3183	Median : 41.948
Mean : 7.015	Mean :0.07714	Mean :-0.3941	Mean : 43.338
3rd Qu.: 8.677	3rd Qu.:0.09346	3rd Qu.:-0.1896	3rd Qu.: 49.101
Max. :12.233	Max. :0.18225	Max. : 1.2671	Max. :137.047

Based on the summary data, it shows that we can't plot the peptide properties in a singular box-plot, also due to them being different scales which have no underlying correlation to each other and are all mathematical predictions. So to check for outliers it would be better to do single boxplots for each of them, due to their differing values/scale-range.

The protein properties would be interesting to see if there is any correlation as they are all physical properties of the molecule itself, but due to the differing value-ranges, it isn't possible to include these in a single boxplot, as such, all the peptide & protein

For ease of use later on, the protein properties will be grouped with the help of a single variable.

```
#Cluster the variables for the protein properties
peptide_data <-(6:9)
protein_data <- (10:13)

# Testing to see if the selections grab the proper data from the datasets.
pander(head(bcel_sars[,peptide_data]))
```

chou_fasman	emini	kolaskar_tongaonkar	parker
1.016	0.703	1.018	2.22
0.77	0.179	1.199	-3.86
0.852	3.427	0.96	4.28
1.41	2.548	0.936	6.32
1.214	1.908	0.937	4.64
0.928	0.547	1.09	0.9

```
pander(head(bcel_sars[,protein_data]))
```

isoelectric_point	aromaticity	hydrophobicity	stability
5.81	0.1033	-0.1438	40.27
6.211	0.06548	-0.0369	25
8.224	0.09179	0.8792	27.86
4.238	0.04478	-0.5214	30.77
6.867	0.1038	-0.5788	21.68
6.867	0.1038	-0.5788	21.68

As we can see the peptide and protein have both been called properly. We can also directly see that the values have different ranges.

As follows, we'll perform boxplots on each value and then differentiate on if it's promoting/non-promoting, to see if there is any concerning outliers and to get a first glimpse at the underlying connection between promoting/non-promoting.

```
# Setup a function to repeatably make the boxplots for the columns
boxplotfun <- function(col_name) {
  ggplot(bcel_sars, aes(x=target, y=!!sym(col_name), fill=target)) + geom_boxplot() +
    theme(plot.title = element_text(size=8), axis.title = element_text(size=8), axis.text = element_tex
    ylab(filter(code_book, Name == col_name)$Label) +
    xlab(NULL) +
    ggtitle(filter(code_book, Name == col_name)$Full.name)
}
# Save the column names to a list for the function
property_names <- c("chou_fasman", "emini", "kolaskar_tongaonkar", "parker", "isoelectric_point", "aroma
# Run the column names through the plot function and catch the plots in a list
plot_finished <- lapply(property_names, boxplotfun)
# Arrange the plots in a neat grid
ggarrange(plotlist = plot_finished, ncol = 2, nrow=4, common.legend = TRUE, legend = "bottom")
```

```
ggsave(
  filename="Figure1.png",
  plot=last_plot(),
  path = "../Data/Images"
)
```

```
## Saving 6.5 x 10 in image
```

Comparing the target values (non-promoting and promoting) with each other, an interesting observation can be made that there isn't much difference at first glance between the non-promoting and promoting peptide sequences. It would be interesting to see where the differences do lie.

A note-able observation is that the emini histogram has become unreadable due to possible outliers. This will be investigated further.

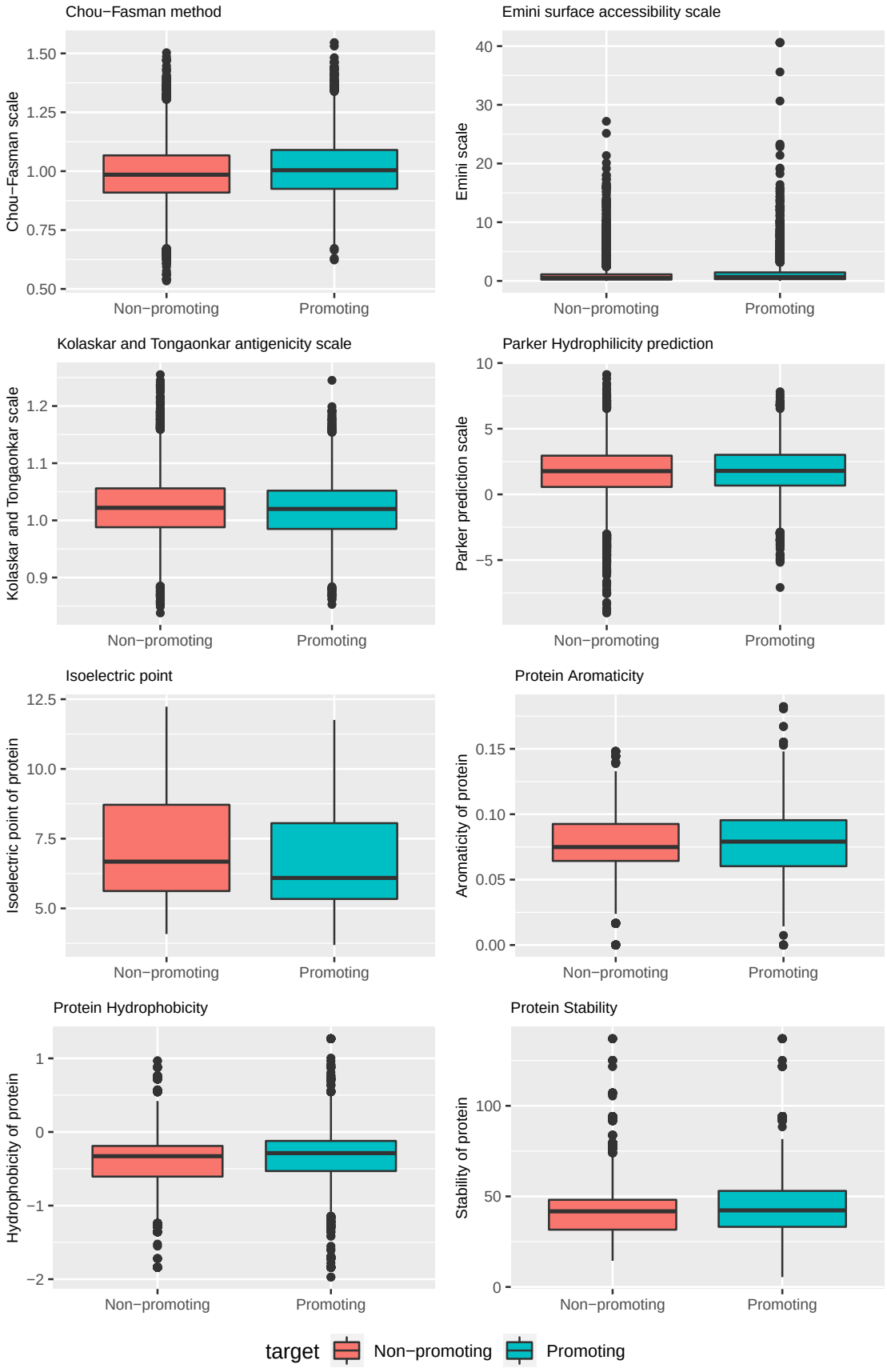


Figure 1: Boxplots of the peptide/protein features

First, a histogram will be made about the Emini surface accessibility scale to get a better view at how bad the outliers stretch outbound due to the boxplot being unreadable.

```
# Make the histogram plot for the emini data
ggplot(bcel_sars, aes(x=emini, fill=target)) + geom_histogram(binwidth=.5) +
  theme(plot.title = element_text(size=8), axis.title = element_text(size=8), axis.text = element_text(s
  ylab(filter(code_book, Name == "emini")$Label) +
  xlab("Emini value") +
  ggtitle(filter(code_book, Name == "emini")$Full.name)
```

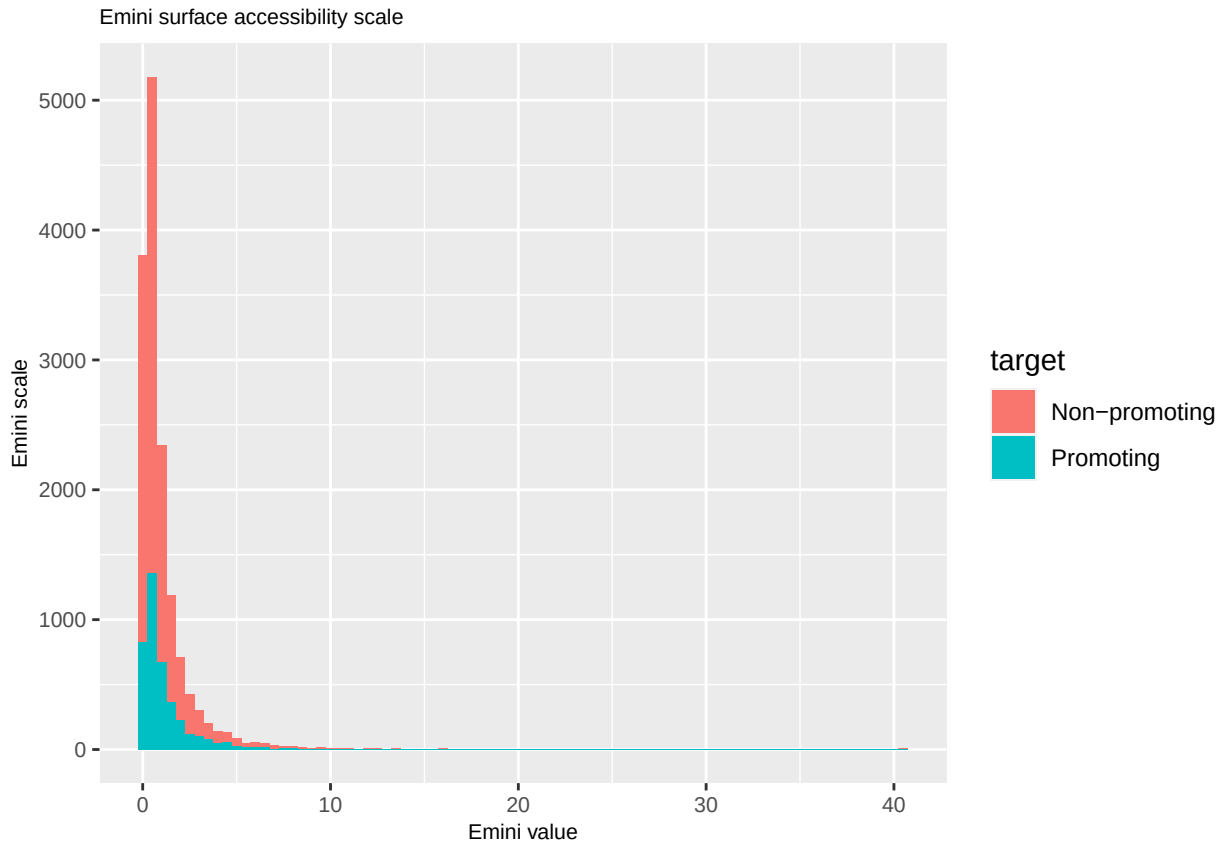


Figure 2: Histogram of the Emini data.

As we can see from the histogram of the Emini surface accessibility scale, the histogram isn't neatly distributed and heavily leans to the left, confirming with the boxplot that there are outliers in the data that should be removed or that the data needs to be log-transformed. Which both will be explored. An interesting second observation is that the there are more peptides that are non-promoting than promoting.

We start with removing the outliers that lie above the upper quartile limit in the emini data.

```
# Calculate the quartiles and the interquartile range
quartiles <- quantile(bcel_sars$emini, probs = c(0.25, 0.75), na.rm=FALSE)
IQR <- IQR(bcel_sars$emini)

# Calculate the upper quantile limit
Upper <- quartiles[2] + 1.5*IQR
```

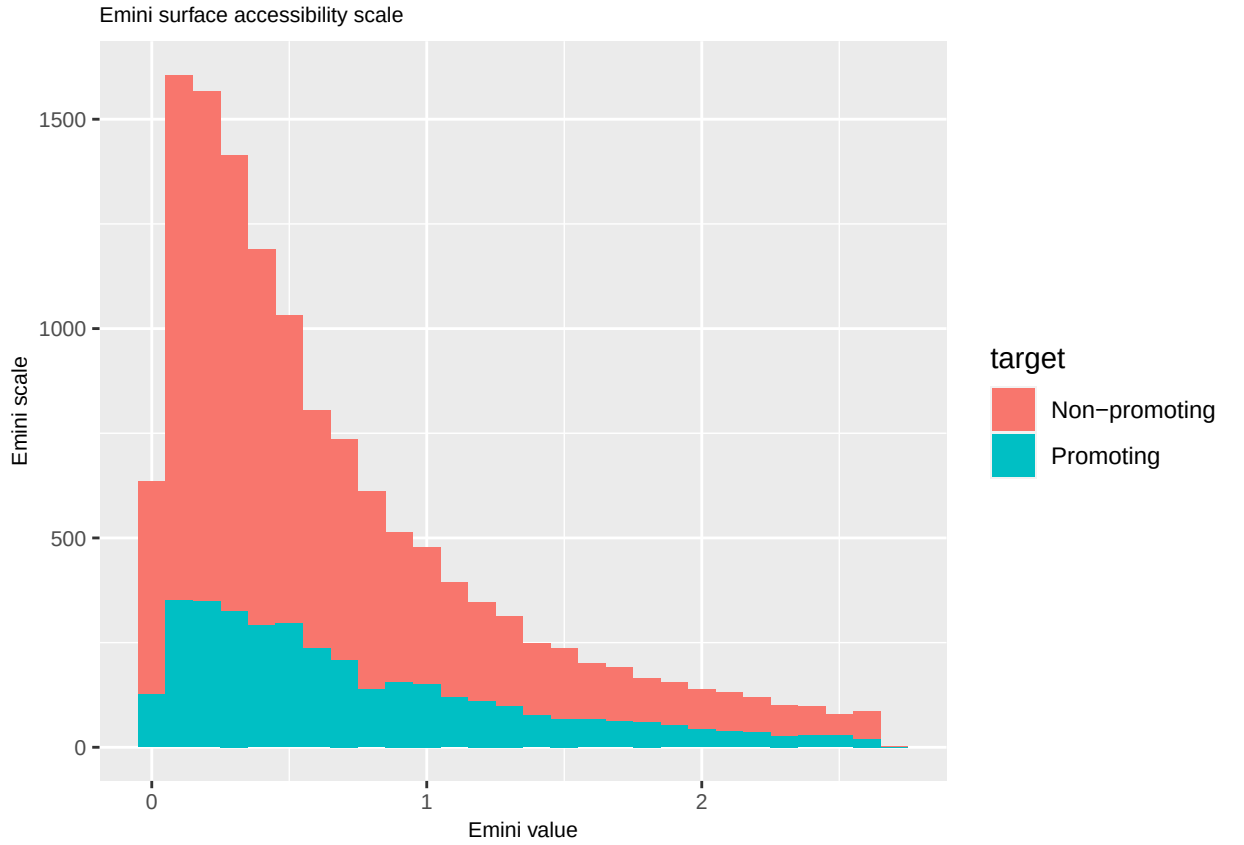
```
#Remove any outlier above the upper quartile limit from the complete dataset
bcel_sars_no_outlier <- subset(bcel_sars, bcel_sars$emini < Upper)
before <- dim(bcel_sars)
after <- dim(bcel_sars_no_outlier)

# Print the deleted amount of outliers.
print(paste("The amount of removed outliers is: ", before[1] - after[1], "outliers"))
```

```
## [1] "The amount of removed outliers is:  1307 outliers"
```

```
#Get the variables out of the memory
remove(quartiles)
remove(IQR)
remove(Upper)
remove(before)
remove(after)
```

The outliers that were removed are those above the upper quartile limit of the emini data. After removing the outliers, it would be wise to print the emini histogram again, to see if this has improved the plot, if there is still a left-leaning histogram and if it's visible that usable data has been cut off unnecessarily.

```
ggplot(bcel_sars_no_outlier, aes(x=emini, fill=target)) + geom_histogram(binwidth=.1) +
  theme(plot.title = element_text(size=8), axis.title = element_text(size=8), axis.text = element_text(s
  ylab(filter(code_book, Name == "emini")$Label) +
  xlab("Emini value") +
  ggtitle(filter(code_book, Name == "emini")$Full.name)
```



In the figure above we can once again see the histogram plot of the emini scale, however, this is the dataset with the removed outliers. As we can see the histogram is still heavily leaning to the left and it still shows that there are a lot more non-promoting than promoting peptides.

What is also visible in the plot is that the data looks cut off for the non-promoting peptides and that we're losing data. As such a log transformation will probably be needed to get a more normal distribution of the data, as we'll do below.

```
#Create a copy of the dataset to transform
bcel_sars_log_transform <- bcel_sars
# Transform the data from the
bcel_sars_log_transform$emini <- log(bcel_sars$emini)
head(bcel_sars_log_transform$emini)
```

```
## [1] -0.3523984 -1.7203695  1.2316852  0.9353087  0.6460556 -0.6033065
```

```
#Checking the data for any non-finite values after a log transformation is usefull.
any(!is.finite(bcel_sars_log_transform$emini))
```

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

Due to there being non-finite values, it's wise to remove those from the dataset, as those can't be worked with.

```
bcel_sars_log_transform <- bcel_sars_log_transform[is.finite(bcel_sars_log_transform$emini),]
before <- dim(bcel_sars)
after <- dim(bcel_sars_log_transform)

print(paste("The amount of removed non-finite numbers is:", before[1] -after[1]))
```

```
## [1] "The amount of removed non-finite numbers is: 5"
```

```
# Remove unused variables from memory
remove(before)
remove(after)
```

After transforming the log data, we'll plot the histogram with the log-transformed data.

```
# Plot the histogram for the log transformed emini data
ggplot(bcel_sars_log_transform, aes(x=emini, fill=target)) + geom_histogram(binwidth=.1) +
  theme(plot.title = element_text(size=8), axis.title = element_text(size=8), axis.text = element_text(s
  ylab(filter(code_book, Name == "emini")$Label) +
  xlab("Emini value") +
  ggtitle(filter(code_book, Name == "emini")$Full.name)
```

```
ggsave(
  filename="Figure2.png",
  plot = last_plot(),
  path = "../Data/Images"
)
```

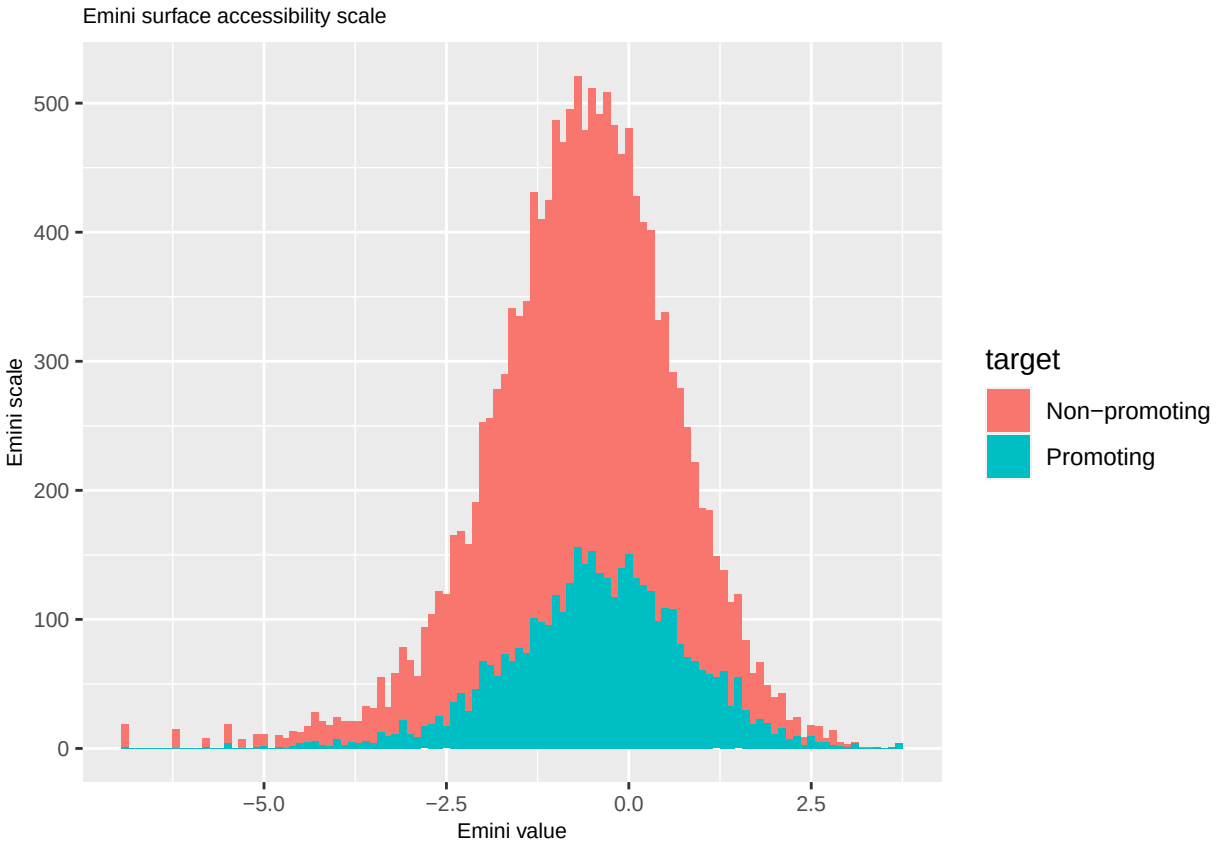


Figure 3: Histogram of the log transformed Emini data

```
## Saving 6.5 x 4.5 in image
```

After log-transforming the emini data, we can see that there is now a semi-decent histogram, showcasing that in regard to removing outliers, log transforming the emini data is the way forward. As this shows that the so-called outliers do play a part in the main data, showcasing that they shouldn't be removed.

We'll plot the boxplot for emini again, but this time with the log-transformed data.

```
# Plot a boxplot for the log-transformed Emini data
ggplot(bcel_sars_log_transform, aes(x=target, y=emini, fill=target)) + geom_boxplot() +
  theme(plot.title = element_text(size=8), axis.title = element_text(size=8), axis.text = element_text(s
  ylab(filter(code_book, Name == "emini")$Label) +
  xlab(NULL) +
  ggtitle(filter(code_book, Name == "emini")$Full.name)
```

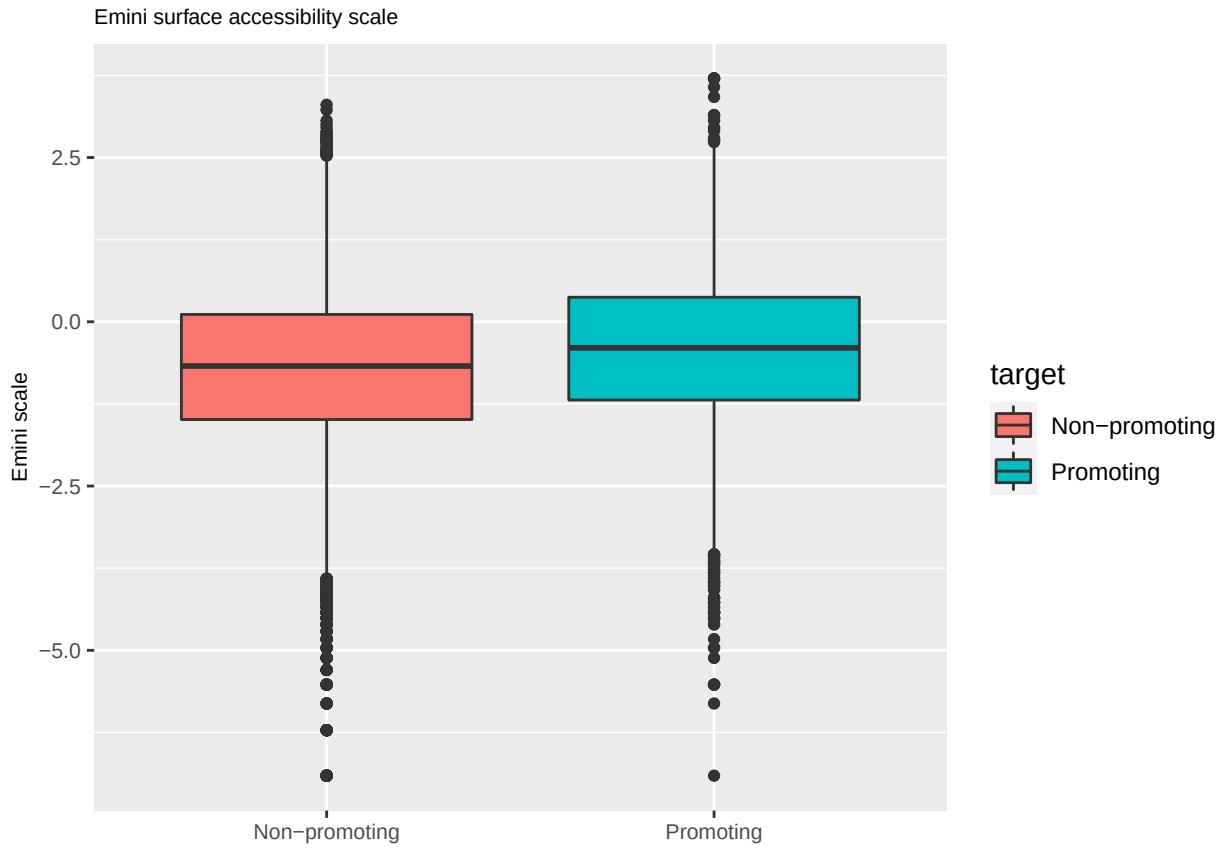


Figure 4: Boxplot of the log-transformed Emini data

As we can see there are still plenty of 'outliers' in the boxplots for the log-transformed data for the emini plots, albeit is the boxplot most certainly more readable and can the non-promoting and promoting data be compared. Upon seeing this, we'll continue using the log transformed information further.

We'll also remake the complete group of boxplots for the report.

```
# Setup a function to repeatably make the boxplots for the columns
boxplotfun <- function(col_name) {
  ggplot(bcel_sars_log_transform, aes(x=target, y=!!sym(col_name), fill=target)) + geom_boxplot() +
```

```
    theme(plot.title = element_text(size=8), axis.title = element_text(size=8), axis.text = element_tex
    ylab(filter(code_book, Name == col_name)$Label) +
    xlab(NULL) +
    ggtitle(filter(code_book, Name == col_name)$Full.name)
}
# Save the column names to a list for the function
property_names <- c("chou_fasman", "emini", "kolaskar_tongaonkar", "parker", "isoelectric_point", "aroma
# Run the column names through the plot function and catch the plots in a list
plot_finished <- lapply(property_names, boxplotfun)
# Arrange the plots in a neat grid
ggarrange(plotlist = plot_finished, ncol = 2, nrow=4, common.legend = TRUE, legend = "bottom")
```

```
ggsave(
  filename="Figure3.png",
  plot=last_plot(),
  path = "../Data/Images"
)
```

```
## Saving 6.5 x 10 in image
```

Multivariate analysis

After analysis and resolving the anomaly that was the emini data in the boxplots, in the end, it was decided to continue with the log-normalized data and continue to determine the correlations between the different properties have been established. The correlations will be calculated with help of a correlation matrix and then plotted into a heatmap. It is expected that the protein & peptide properties will be correlated to each other.

```
#Make a correlation matrix for the peptide/protein properties
res <- cor(bcel_sars_log_transform[,6:13])
res
```

```
##                     chou_fasman         emini kolaskar_tongaonkar       parker
## chou_fasman          1.00000000  0.3516465828         -0.44436872  0.590823972
## emini                0.35164658  1.0000000000         -0.59770892  0.611048723
## kolaskar_tongaonkar -0.44436872 -0.5977089228          1.00000000 -0.688766999
## parker               0.59082397  0.6110487229         -0.68876700  1.000000000
## isoelectric_point    0.01306379 -0.0065203980         -0.01499174 -0.008491778
## aromaticity          0.00705453 -0.0002637006          0.12331412 -0.185794412
## hydrophobicity      -0.13915702  0.0073699345          0.30039277 -0.300469191
## stability            0.11347695 -0.0578307386         -0.03149296  0.082413628
##                     isoelectric_point   aromaticity hydrophobicity   stability
## chou_fasman              0.013063788  0.0070545301   -0.139157019  0.11347695
## emini                   -0.006520398 -0.0002637006    0.007369934 -0.05783074
## kolaskar_tongaonkar     -0.014991742  0.1233141239    0.300392771 -0.03149296
## parker                  -0.008491778 -0.1857944120   -0.300469191  0.08241363
## isoelectric_point        1.000000000 -0.2140864360   -0.210519246  0.21365232
## aromaticity             -0.214086436  1.0000000000    0.322740997 -0.25636557
## hydrophobicity          -0.210519246  0.3227409973    1.000000000 -0.40902633
## stability                0.213652324 -0.2563655731   -0.409026334  1.00000000
```

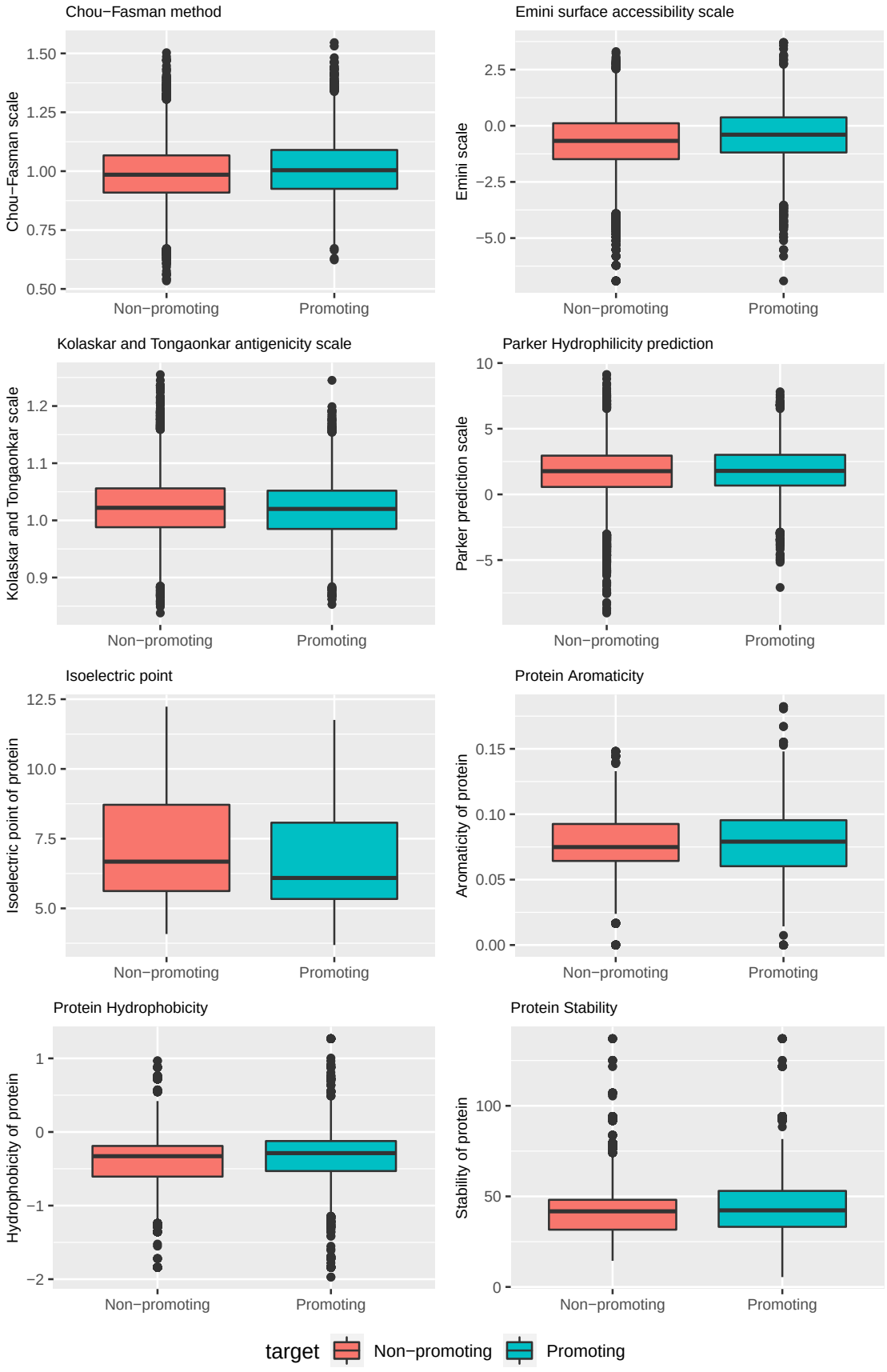


Figure 5: Boxplots of all data with log-transformed Emini data

As we can see the correlation matrix has been created successfully and will now be plotted as a heatmap.

```
# save the heatmap for the correlation matrix
png("../Data/Images/Figure4_heatmap.png",
    width =720,
    height =480)
plot.new()
heatmap(x = res, scale="column", col = heat.colors(256),
        main = "Relation heatmap of the different protein/peptide properties")
dev.off()
```

```
## pdf
##   2
```

```
#Print the heatmap so that it shows up in the PDF of the EDA_log
heatmap(x = res, scale="column", col = heat.colors(256),
        main = "Relation heatmap of the different protein/peptide properties")
```

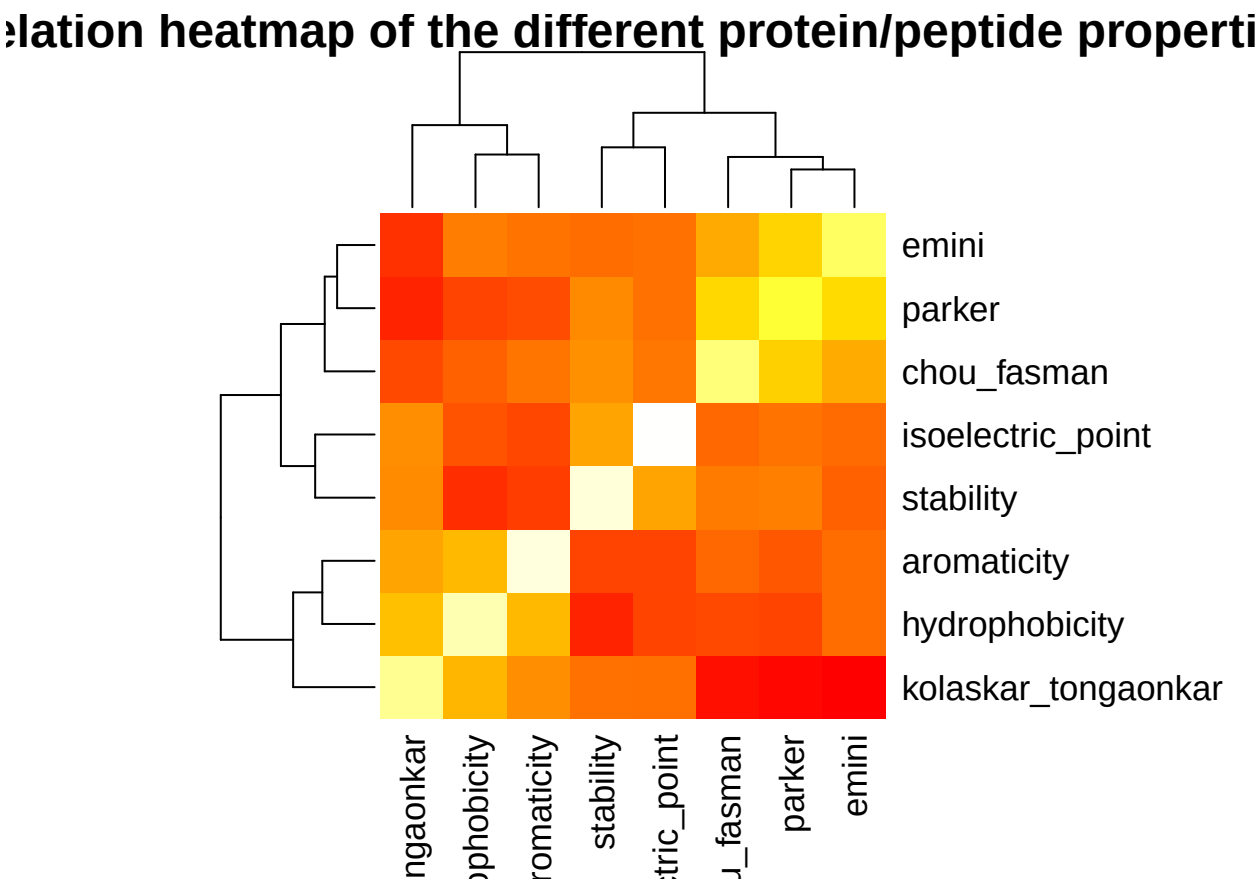


Figure 6: Heatmap of the correlation between the properties

As we can see from the plotted correlation matrix heatmap, some aspects of the data are indeed related to each other, while some other aspects, surprisingly aren't. This might be due to the different scales and ranges that are being used for each of the properties.

Now that we've looked at the correlations between the properties, it's time to perform a Principal Component Analysis to visualize the variation in the dataset.

```
# Calculate the PCA
bcel_sars.pca <- prcomp(bcel_sars_log_transform[,6:13], center = TRUE, scale. = TRUE)
# Show a summary of the PCA
summary(bcel_sars.pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6     PC7
## Standard deviation     1.6759 1.3225 0.9280 0.9018 0.79085 0.76809 0.55908
## Proportion of Variance 0.3511 0.2186 0.1076 0.1017 0.07818 0.07375 0.03907
## Cumulative Proportion  0.3511 0.5697 0.6774 0.7790 0.85718 0.93093 0.97000
##                           PC8
## Standard deviation     0.4899
## Proportion of Variance 0.0300
## Cumulative Proportion  1.0000
```

After calculating the PCA, the PCA will be plotted with the help of the ggbiplot package.

```
ggbiplot(bcel_sars.pca, groups = bcel_sars_log_transform$target, varname.size=4, size=0.001) +
        scale_color_manual(name="Antigenicity", values=c("yellow", "palegreen"))
```

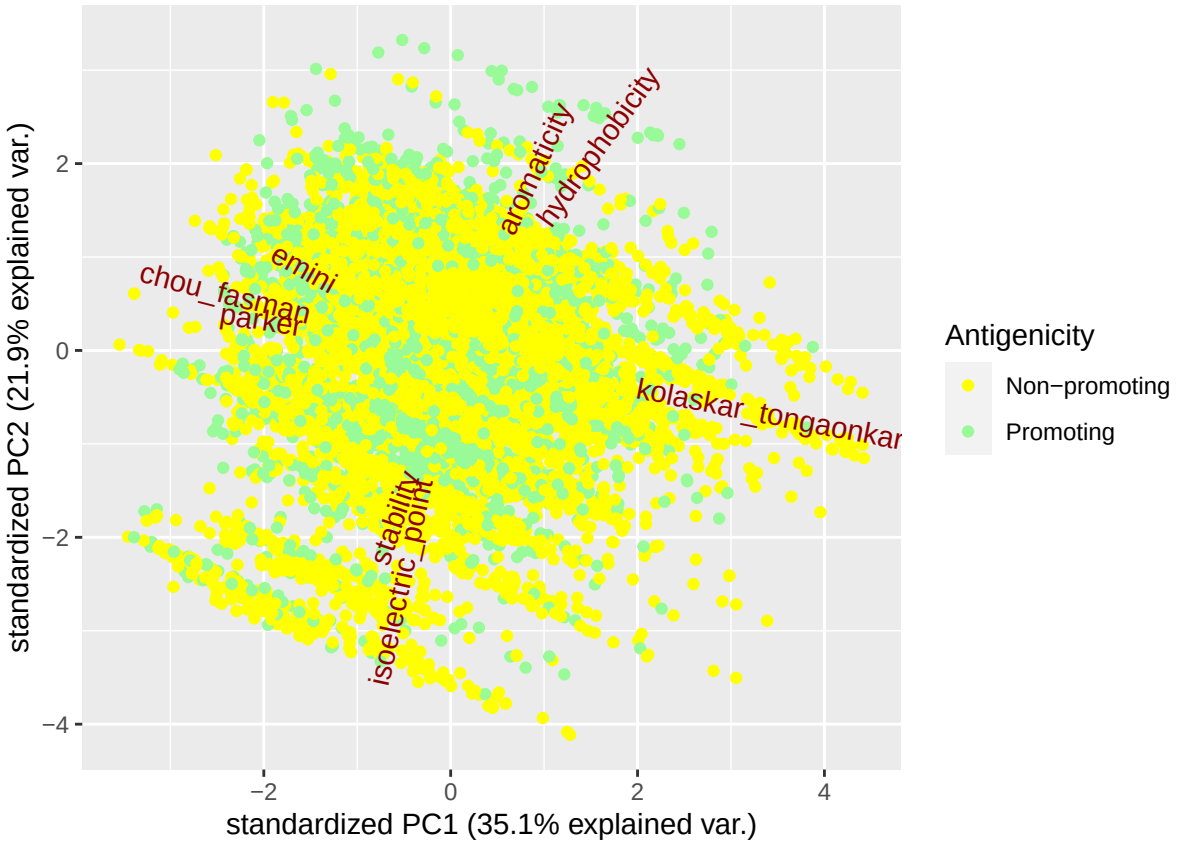


Figure 7: PCA plot of the protein/peptide properties

As we can see from the PCA is that some variables are closely interlinked, forming 4 different groups. These being: 1. Chou_fasman, Parker & Emini 2. Aromaticity, hydrophobicity 3. stability, isoelectric_point 4. Kolaskar_tongaonkar

And further we can see that there aren't really any clear differences between promoting and non-promoting peptide sequences.

```
ggsave(
  filename="Figure5.png",
  plot = last_plot(),
  path = "../Data/Images"
)
```

```
## Saving 6.5 x 4.5 in image
```

Export data

After this, we'll export the data so that it can be used during machine learning, for this we'll export the normal data & the log transformed dataset.

```
#Write the data files to a .arff format for use in Weka
bcel_sars_export <- bcel_sars[,6:14]
bcel_sars_log_transform_export <- bcel_sars_log_transform[,6:14]
write.arff(bcel_sars_export, file="../Data/bcel_sars_data.arff")
write.arff(bcel_sars_log_transform_export, file="../Data/bcel_sars_log_transform.arff")
```