Appendix 2 - Linear model and cluster analysis of Influence factors and Public Transport subscriptions

Gabriel Peier

2022-12-22

1 INTRODUCTION

In this script, the methodical part of the Linear Model and the Cluster Analysis is described and executed.

1.1 Packages

For this analysis, several packages for the coding were needed. To make the use and reproducability easier, I list them here at the beginning of the paper. I did use the following packages:

```
library(tidyverse)              # ggplot2, dplyr, tidyr, readr, tibble
library(stringr)
library(dplyr)
library(caret)                  # data splitting and pre-processing
library(PerformanceAnalytics)   # special graphical comparisons of variables
library(ggpubr)                 # for using ggbarplot
library(regclass)               # For VIF function (Variance Inflation Factor)
library(MASS)
library(reshape2)               # for function "melt" => combine with ggplot
library(cluster)                # for silhouette plot function (cluster analysis)
library(dbscan)                 # for DB Scan Clustering
library(mclust)                 # for Gaussian mixture model
library(dbscan)                 # DBscan model (clustering)
library(plyr)                   # To combine dataframes
library(GGally)                 # for ggpairs plot
```

1.2 Loading data

```
getwd()
```

```
## [1] "G:/My Drive/MasterThesis/Scripts"
```

```
d.share <- read.csv(("../Data/3_Output/inf_fac_share.csv"))
d.count <- read.csv(("../Data/3_Output/inf_fac_count.csv"))
```

1.3 NA handling

In the dataset exists NaN values, which can not be handled within a lm function (in contrast to NA value). So the values have to be replaced:

```
# Showing NA values per Column
apply(is.na(d.share), 2, sum)
```

```
##              BFS_Nr        municipality              canton            language
##                   0                   0                   0                   0
##       pop_count_BFS        single_share       married_share       widowed_share
##                   0                   1                   1                   1
##      divorced_share            GA_share           HTA_share           FNT_share
##                   1                   1                   1                   1
##             age0_20            age20_40            age40_60              age60.
##                  11                  11                  11                  11
##         birth_munic          birth_cant            birth_CH         birth_notCH
##                  11                  11                  11                  11
##                male              female          resid_0_1y          resid_1_5y
##                  11                  11                  11                  11
##         resid_6_10y          resid_10.y               hh_1               hh_2
##                  11                  11                  11                  11
##              hh_3_5               hh_6.      PT_dist_medium      PT_time_medium
##                  11                  11                  34                  34
##         PT_dist_big         PT_time_big     str_dist_medium     str_time_medium
##                  34                  34                  34                  34
##        str_dist_big        str_time_big         PT_fact_big      PT_fact_medium
##                  34                  34                  34                  34
##   bus_stops_per_pop train_stops_per_pop other_stops_per_pop   bus_stat_per_1000
##                  67                  67                  67                  67
## train_stat_per_1000 other_stat_per_1000       comb_car_1000         el_car_1000
##                  67                  67                  11                  11
##       inbound_share      outbound_share
##                 133                 133
```

Most missing values are visible in the inbound and outbound share. This comes due to the old data from 2000. Many municipalities have changed since then. To not loose all these information, the mean value will be replacing the share values in these 2 cases:

```
# replacing mean value in inbound and outbound share data:
d.share$inbound_share[is.na(d.share$inbound_share)] <- mean(d.share$inbound_share, na.rm=TRUE)
d.share$outbound_share[is.na(d.share$outbound_share)] <- mean(d.share$outbound_share, na.rm=TRUE)
```

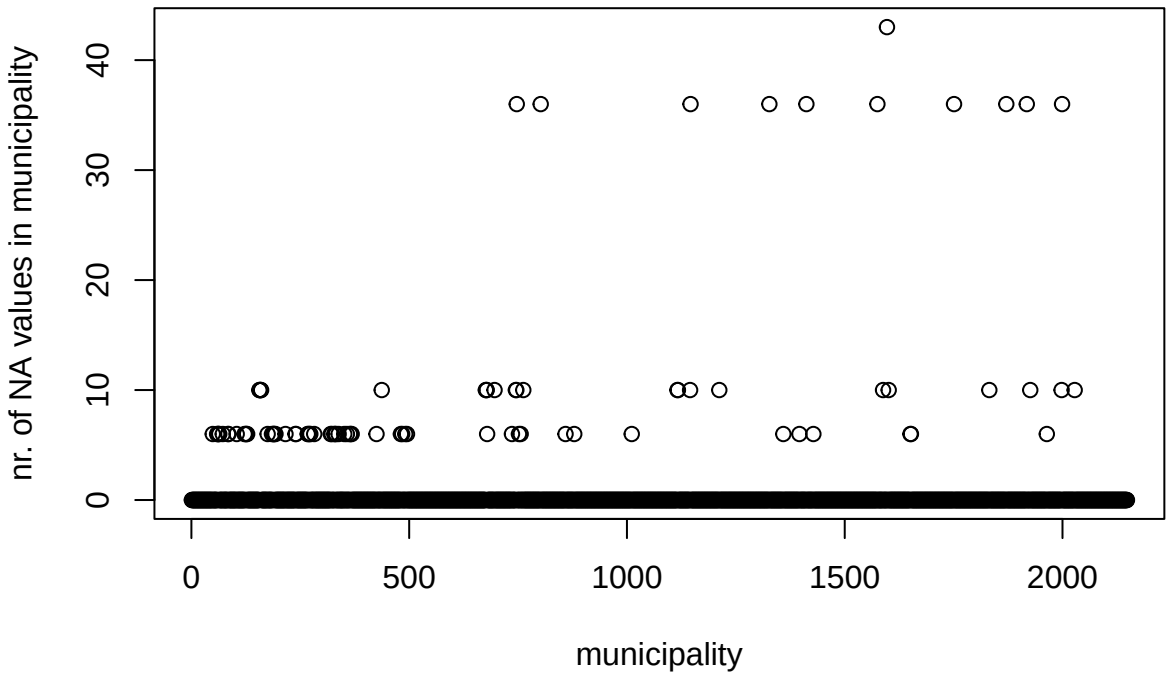
Second, most missing data are in the stops and station tables (67). Let's have a look at the number of rows, where still data is missing:

```
sum(!complete.cases(d.share))
```

```
## [1] 90
```

Beside the 67 missing from the stops table, only 23 observations more do lack data. So there are many rows with multiple values missing. This can be shown graphically:

```
plot(rowSums(is.na(d.share)), xlab = "municipality", ylab="nr. of NA values in municipality")
```



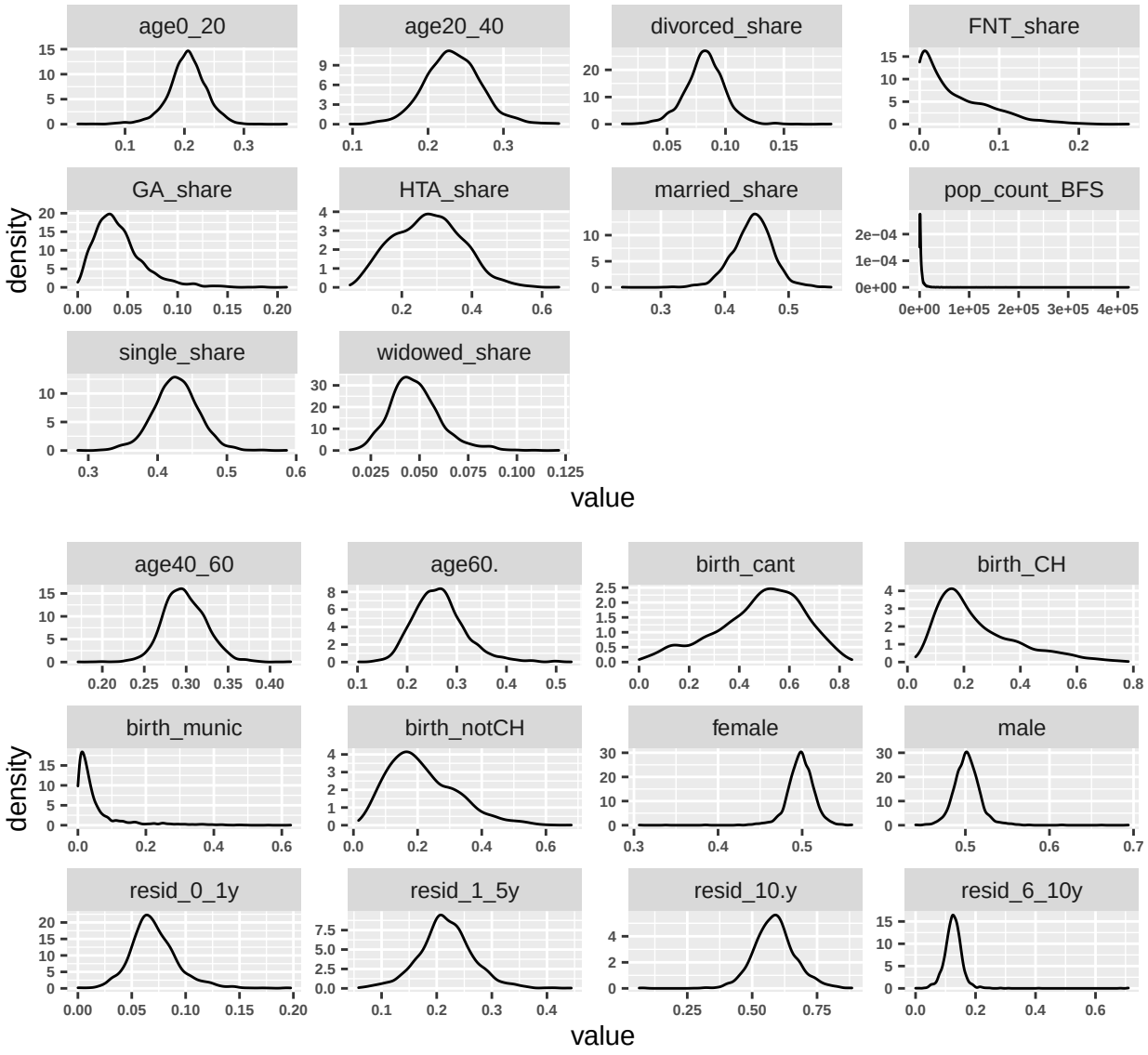
11 observations have 36 (or more) missing values. This reflects all columns, which show at least 11 missing values. For these specific municipalities, it doesn't make any sense to compute the mean value. Also for the others, there are always multiple columns with missing data at the same time (at least 6), so these rows will be deleted. Still enough data is there.

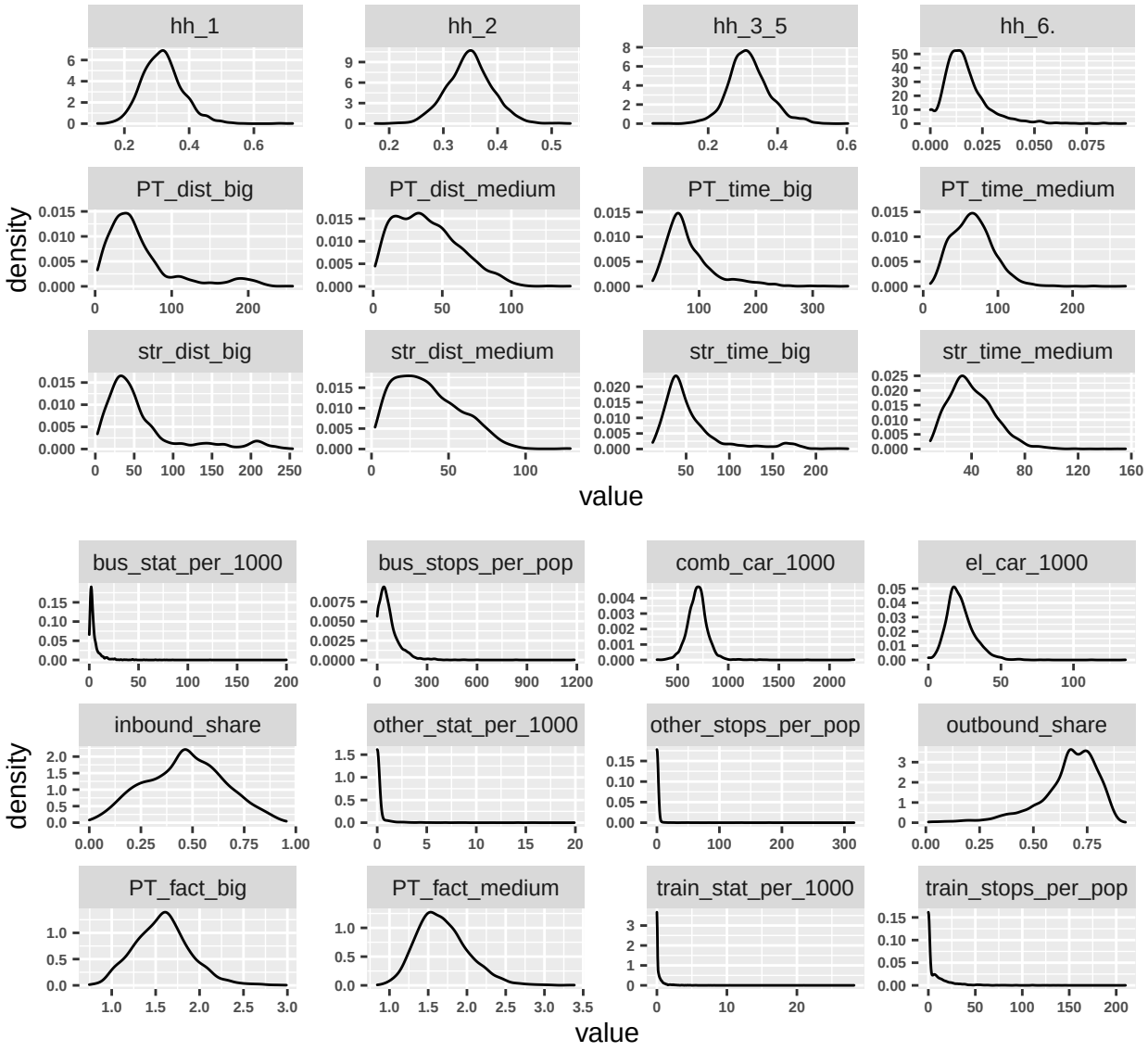
```
# replace NaN values or "Inf" values with NA
d.share[is.na(d.share) | d.share == "Inf"] <- NA
d.count[is.na(d.count) | d.count == "Inf"] <- NA

## Removing all observations with NA!

d.share <- na.omit(d.share)
d.count <- na.omit(d.count)
```

1.4 Graphical Analysis

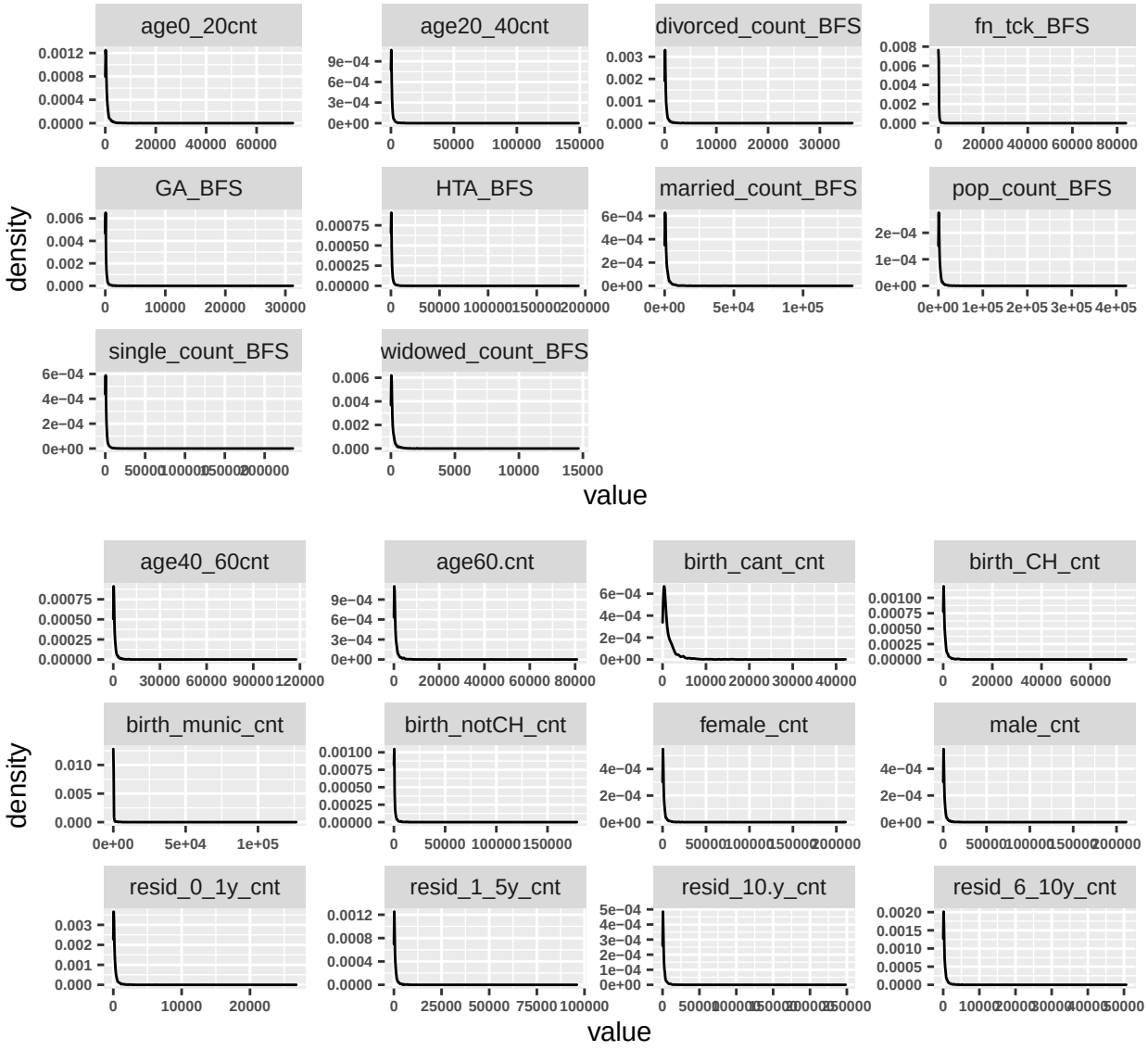
In this chapter, several analysis steps based on graphics will be performed to get a meaningful insight into the data. The used methods only cover a little part of the huge possibilities when it comes to visualizations. I mainly focused on the "ggplot"-package as it comes with a wide range of options.

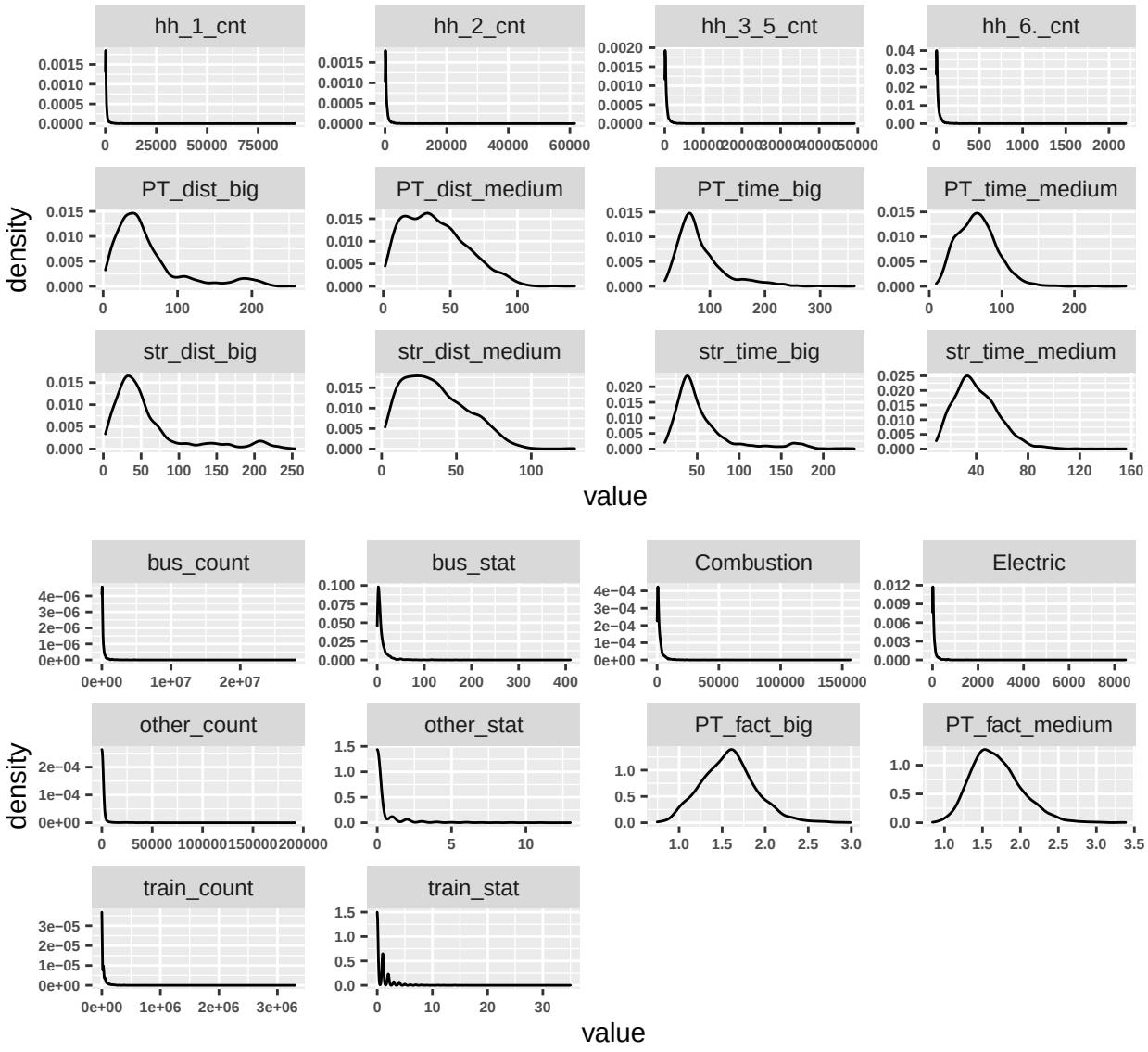
1.4.1 Share Data: Density plots of different influence factors




hh_1
hh_2
hh_3_5
hh_6.
PT_dist_big
PT_dist_medium
PT_time_big
PT_time_medium
str_dist_big
str_dist_medium
str_time_big
str_time_medium
density
value
bus_stat_per_1000
bus_stops_per_pop
comb_car_1000
el_car_1000
inbound_share
other_stat_per_1000
other_stops_per_pop
outbound_share
PT_fact_big
PT_fact_medium
train_stat_per_1000
train_stops_per_pop
density
value

1.4.2 Count Data: Density plots of different influence factors





All data except the PT_fact_big and PT_fact_medium are right-skewed and have to be logarithmized to normalize!

1.5 Normalizing data

The normalization is important due to some outstanding variables. Due to the different data distribution in the share and count datasets, different approaches will be used for the two sets.

1.5.1 Normalizing Share Values

In the case of the share dataset, the normalization mainly affects those variables that have no shares, as these already lie between 0 and 1 anyway. But to standardize all continuous variables at the same scale, I will include all parameters except the target variable into the normalization. The 'scale'-function should do the necessary normalization in these cases.

```r
# NORMALIZING SHARE VALUES
d.norm.share <- d.share[,4:50]
# print(colnames(d.norm.share))

# define columns to normalize:
columns <- c("single_share", "married_share",
             "widowed_share", "divorced_share", "age0_20", "age20_40", "age40_60",
             "age60.", "birth_munic", "birth_cant", "birth_CH", "birth_notCH",
             "male", "female", "resid_0_1y", "resid_1_5y", "resid_6_10y", "resid_10.y",
             "hh_1", "hh_2", "hh_3_5", "hh_6.", "PT_dist_medium", "PT_time_medium",
             "PT_dist_big", "PT_time_big", "str_dist_medium", "str_time_medium",
             "str_dist_big", "str_time_big", "PT_fact_big", "PT_fact_medium",
             "bus_stops_per_pop", "train_stops_per_pop", "other_stops_per_pop",
             "bus_stat_per_1000", "train_stat_per_1000", "other_stat_per_1000",
             "comb_car_1000", "el_car_1000", "inbound_share", "outbound_share")

# normalize data!
d.norm.share[columns] <- scale(d.norm.share[columns])
  # scale function: mean = 0, standard deviation = 1
```

The used scale function normalizes data through centering (setting mean of each variable to 0 by subtracting the mean value of all observations) and scaling (setting standard deviation to 1 by dividing all observations to the standard deviation of the column). The values achieved in this way all have the same weight in relation to a model. Thus, the resulting parameters can be compared with each other later on.

1.5.2 Normalizing Count data

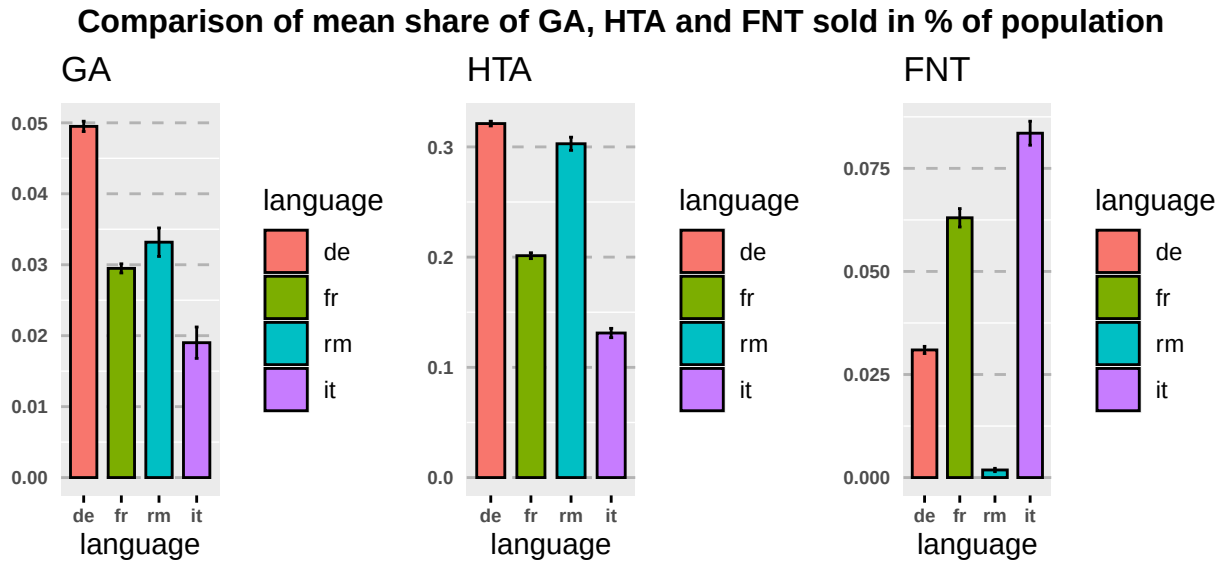
As observed in chapter 1.4.2, most count data have a right-skewed distribution and should therefore be log-transformed. The only variables this hardly affects are "PT_fact_big" and "PT_fact_medium", which have an approximate normal distribution. For this reason, these variables are not taken into account in the normalization.

```r
# NORMALIZING COUNT DATA

# log-transform all columns except PT_fact_big, PT_fact_medium and language
d.norm.count <- log(d.count[, !names(d.count)
                            %in% c("language", "PT_fact_big", "PT_fact_medium",
                                   "BFS_Nr", "municipality", "canton")]+1)
                                     # '+1' due to log(0) = -Inf!

# add the 3 deleted columns again!
d.norm.count["language"] <- d.count["language"]
d.norm.count["PT_fact_big"] <- d.count["PT_fact_big"]
d.norm.count["PT_fact_medium"] <- d.count["PT_fact_medium"]
```

1.6 GA, HTA & FNT shares compared to languages:

A first insight can be gained by comparing the different target share values (HTA, GA, FNT) with the language regions as the only categorical variable present:

```
plot_HTA <- ggbarplot(d.share, x="language", y="HTA_share", fill="language",
                      add = "mean_se", xlab = "language", legend = "right",
                      title = "HTA", ggtheme = theme_cleveland()) + # better looking
  theme(axis.text=element_text(size=6.5, face="bold"))

plot_GA <- ggbarplot(d.share, x="language", y="GA_share", fill="language", add = "mean_se",
                      xlab = "language", legend = "right",
                     title = "GA", ggtheme = theme_cleveland()) +
  theme(axis.text=element_text(size=6.5, face="bold"))

plot_FNT <- ggbarplot(d.share, x="language", y="FNT_share", fill="language",
                      add = "mean_se", xlab = "language", legend = "right",
                      title = "FNT", ggtheme = theme_cleveland()) +
  theme(axis.text=element_text(size=6.5, face="bold"))

share.comparison <- ggarrange(plot_GA, plot_HTA, plot_FNT, ncol=3) # 3 plots beside each other

annotate_figure(share.comparison,
    top = text_grob("Comparison of mean share of GA, HTA and FNT sold in % of population",
                                face = "bold")) # set over-all title ("top")
```



Interesting insights can be gained here: For the GA tickets, the share is highest in the German municipalities and lowest in the Italian regions. The rate in the German-speaking region is thereby 2.5 times as high as in Ticino. The picture is similar for the Half-Fare Card (HTA), with shares generally in much higher percentage ranges. The relatively high proportion in Rhaeto-Romanic speaking municipalities is remarkable, which is about the same as in German-speaking Switzerland.

A completely different picture emerges for the fare network tickets (FNT). Now, the non-German-speaking areas, led by Ticino with just under 8%, have the highest share of fare network tickets. The rate in French-speaking Switzerland is also not much lower at around 6%, while German-speaking Switzerland only achieves a share of 3%.

2 MODELLING SHARES

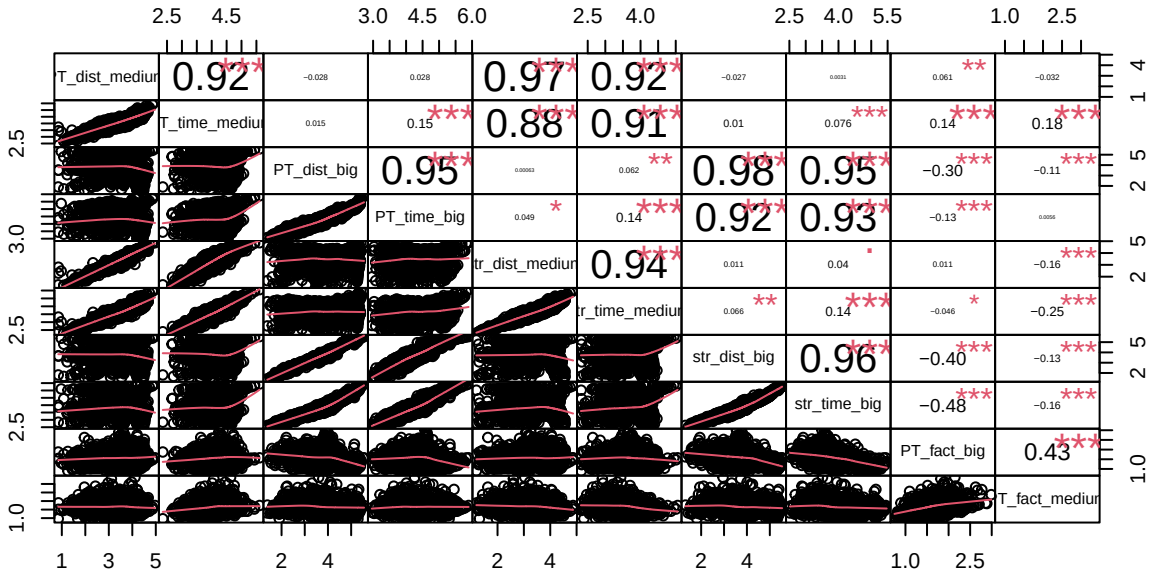
I will achieve the first objective of modelling the share of season tickets through two different approaches: Linear model (LM) and generalized linear model (GLM) with family "binomial".

2.1 Parameter selection

Before starting with the models, the amount of possible influence factors must be reduced. Still many parameters are present, which strongly rely on each other, for example the different PT / Street distance + time tables:

2.1.1 street and PT distance + time tables:

```
chart.Correlation(d.norm.count[,c(27:34, 44:45)], # +1 due to 0-values (log(0) = -Inf)
                  histogram=FALSE) # adding histograms to the plot
```



Many correlation values are strongly significant, the collinearity is huge in these cases! There has to be done something here. The distance and time data are always strongly correlating for big and medium cities, so I will focus only on time data. Within the same city category, PT and str data are additionnally also correlating highly, so only the PT data is used. The PT factor will be used as well, the correlation values are not that high, even if one of the inputs for the calculation are the PT_time_big and PT_time_medium tables. Decision: For further uses, the following variables will be used:

- PT_fact_big
- PT_fact_medium
- PT_time_big
- PT_time_medium

```
# remove undesired columns #1 from COUNT DATA
d.norm.count <- d.norm.count[, !names(d.norm.count) # remove the following columns:
                             %in% c("PT_dist_medium", "PT_dist_big", "str_dist_medium",
                                    "str_time_medium", "str_dist_big", "str_time_big")]

# remove undesired columns #1 from SHARE DATA
d.norm.share <- d.norm.share[, !names(d.norm.share) # remove the following columns:
                             %in% c("PT_dist_medium", "PT_dist_big", "str_dist_medium",
                                    "str_time_medium", "str_dist_big", "str_time_big")]
```

2.1.2 Categorical data

Furthermore, many original categorical data (personal data) has been used to show demographic structure (age segments, household size etc.). With the given population data (pop_count_BFS) and the categories male and female for example, the number of male explains the number of females in the municipality (population - male). For each of this categories, one can be removed. This affects the following variables, which are removed:

- divorced_count_BFS
- age60.cnt
- birth_notCH_cnt
- female_cnt
- resid_10.y_cnt
- hh6._cnt

The same procedure can also be used for the share data set, since, for example, the proportion of men in a municipality simultaneously explains the proportion of women.

```
# remove undesired columns #2 SHARE DATA
d.norm.count <- d.norm.count[, !names(d.norm.count) # remove the following columns:
                             %in% c("divorced_count_BFS", "age60.cnt", "birth_notCH_cnt",
                                    "female_cnt", "resid_10.y_cnt", "hh_6._cnt")]

# remove undesired columns #2 COUNT DATA
d.norm.share <- d.norm.share[, !names(d.norm.share) # remove the following columns:
                             %in% c("divorced_share", "age60.", "birth_notCH",
                                    "female", "resid_10.y", "hh_6.")]
```

2.2 Train / Test splitting Share/count datasets (original + normalized each)

With the selection of data complete, the data set can now be split into a training and test set. This is important in modelling to detect potential overfitting if the accuracy of the model is much higher for the training set than for the test data. Splitting is done for all data sets that will be used later in the modelling, i.e. the normalised count and share data:

2.2.1 Normalized share dataset

```
set.seed(234) # reproducible
indexes <- createDataPartition(d.norm.share$pop_count_BFS, p = .8, list = FALSE)
```

```
d.norm.share.train <- d.norm.share[indexes, ]
d.norm.share.test <- d.norm.share[-indexes, ]
```

2.2.2 Normalized count dataset

```
set.seed(234) # reproducible
indexes <- createDataPartition(d.norm.count$pop_count_BFS, p = .8, list = FALSE)
d.norm.count.train <- d.norm.count[indexes, ]
d.norm.count.test <- d.norm.count[-indexes, ]
```

2.3 Modelling General Season tickets (GA)

This analysis starts with the first target variable, the GA. The share as well as the count will be calculated in the following chapter using two methods, the Generalized Linear Model with family Binomial and the Linear Model.

2.3.1 Generalized Linear model (GLM) with family "Binomial" (GA)

With a linear model, no probability distributions can be predicted, as values above 1 and below 0 are possible, resulting in meaningless values concerning the share. Therefore, a logistic approach comes into play, which strictly forecasts values between 0 and 1. Although I do not have values for individuals with regard to the share and thus do not model a binary target variable, a generalized linear model with family binomial as adequate, as binomial data are per definition strictly between 0 and 1.

2.3.1.1 Binomial vs. quasibinomial

There are several options when choosing the option "family" in the GLM. By default, "binomial" is used for a binomial model. In cases where the dispersion is too large or too small (over- and underdispersion), a "quasibinomial" model can be used, which has an additional dispersion parameter. This helps to ensure that significances can be detected. For these reasons, a quasibinomial model is also used here. The disadvantage here is that no stepwise procedure for paremeter elimination can be applied, as this is not permitted by the definition of the model.

2.3.1.2 GLM model runs (GA)

The first run is done by using all possible influence factors except the two remaining target variables:

```
#####  Modelling GLM with share data:

################### Model set up 01 ######################
glm.share.01 <- glm(GA_share ~ (. - HTA_share - FNT_share)^2, # all interactions included!
                family = quasibinomial, data=d.norm.share)

summary(glm.share.01)
### Summary output not printed, due to all possible interactions shown
```

Starting from the output of the first experiment, all possible interactions outside the script were examined and all those that were classified as relevant were included in the second experiment, while all other interactions were no longer present:

```
######## Including interaction (see excel sheet)
glm.share.02 <- glm(GA_share ~ . - HTA_share - FNT_share
                    + bus_stops_per_pop:bus_stat_per_1000
                    + train_stops_per_pop:train_stat_per_1000
                    + other_stops_per_pop:other_stops_per_pop
                    + PT_fact_big:PT_fact_medium
                    + PT_time_big:PT_fact_big
                    + PT_time_medium:PT_time_big
                    + PT_fact_medium:PT_time_medium,
                family = binomial, data=d.norm.share)
    ## binomial model as a try for further stepwise selection!
summary(glm.share.02)
```

```
Call:
glm(formula = GA_share ~ . - HTA_share - FNT_share + bus_stops_per_pop:bus_stat_per_1000 +
    train_stops_per_pop:train_stat_per_1000 + other_stops_per_pop:other_stops_per_pop +
    PT_fact_big:PT_fact_medium + PT_time_big:PT_fact_big + PT_time_medium:PT_time_big +
    PT_fact_medium:PT_time_medium, family = binomial, data = d.norm.share)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.39359  -0.06229  -0.00999   0.04674   0.43832

Coefficients:
                                          Estimate Std. Error z value Pr(>|z|)
(Intercept)                             -3.080e+00  1.987e-01 -15.505   <2e-16 ***
languagefr                              -3.418e-01  4.086e-01  -0.836    0.403
languageit                              -1.101e+00  1.132e+00  -0.972    0.331
languagerm                              -4.106e-01  9.793e-01  -0.419    0.675
pop_count_BFS                            7.259e-07  8.349e-06   0.087    0.931
single_share                             6.633e-03  3.457e-01   0.019    0.985
married_share                           -1.944e-02  3.047e-01  -0.064    0.949
widowed_share                           -1.448e-03  2.357e-01  -0.006    0.995
age0_20                                 -9.213e-02  2.667e-01  -0.345    0.730
age20_40                                -4.930e-02  2.200e-01  -0.224    0.823
age40_60                                -1.949e-02  1.700e-01  -0.115    0.909
birth_munic                              1.570e-01  2.019e-01   0.777    0.437
birth_cant                               2.209e-01  3.289e-01   0.672    0.502
birth_CH                                 1.902e-01  3.007e-01   0.633    0.527
male                                    -5.925e-02  1.367e-01  -0.433    0.665
resid_0_1y                               2.243e-02  1.534e-01   0.146    0.884
resid_1_5y                               2.638e-02  1.726e-01   0.153    0.878
resid_6_10y                              2.080e-02  1.093e-01   0.190    0.849
hh_1                                     3.040e-01  8.657e-01   0.351    0.726
hh_2                                     1.640e-01  6.056e-01   0.271    0.787
hh_3_5                                   2.364e-01  8.345e-01   0.283    0.777
PT_time_medium                          -1.032e-01  1.373e-01  -0.751    0.452
PT_time_big                             -1.243e-01  2.494e-01  -0.499    0.618
PT_fact_big                             -1.708e-01  1.600e-01  -1.068    0.286
PT_fact_medium                           2.558e-03  1.401e-01   0.018    0.985
bus_stops_per_pop                       -2.274e-02  1.460e-01  -0.156    0.876
train_stops_per_pop                      9.010e-02  2.038e-01   0.442    0.658
other_stops_per_pop                     -1.797e-02  1.105e-01  -0.163    0.871
bus_stat_per_1000                        1.458e-01  2.127e-01   0.685    0.493
train_stat_per_1000                     -5.543e-02  3.252e-01  -0.170    0.865
other_stat_per_1000                      3.691e-02  1.267e-01   0.291    0.771
comb_car_1000                           -8.249e-02  1.495e-01  -0.552    0.581
el_car_1000                              3.670e-02  1.419e-01   0.259    0.796
inbound_share                            6.851e-03  1.602e-01   0.043    0.966
outbound_share                           4.453e-02  1.668e-01   0.267    0.790
bus_stops_per_pop:bus_stat_per_1000     -2.835e-03  2.164e-02  -0.131    0.896
train_stops_per_pop:train_stat_per_1000 -3.874e-06  3.191e-02   0.000    1.000
PT_fact_big:PT_fact_medium               3.552e-02  1.131e-01   0.314    0.753
PT_time_big:PT_fact_big                  8.413e-03  1.807e-01   0.047    0.963
PT_time_medium:PT_time_big               4.287e-02  7.507e-02   0.571    0.568
PT_time_medium:PT_fact_medium           -2.517e-02  1.431e-01  -0.176    0.860
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 30.792  on 2057  degrees of freedom
Residual deviance: 14.849  on 2017  degrees of freedom
AIC: 257.43

Number of Fisher Scoring iterations: 7
```

Now a model with the family "binomial" was used. It is relatively quickly apparent that no parameter is considered significant. However, in principle, a stepwise procedure for parameter selection is possible in this way, which is applied in the next step:

```
############### Stepwise parameter selection forward and backwards!###########
glm.share.03 <- stepAIC(glm.share.02, direction="both", trace=FALSE)

summary(glm.share.03)
```

```
##
## Call:
## glm(formula = GA_share ~ age0_20 + PT_time_big + PT_fact_big,
##     family = binomial, data = d.norm.share)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.29242  -0.07647  -0.01404   0.05599   0.53501
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.1860     0.1156 -27.560   <2e-16 ***
## age0_20      -0.2025     0.1266  -1.600   0.1096
## PT_time_big  -0.2456     0.1300  -1.889   0.0589 .
## PT_fact_big  -0.2013     0.1214  -1.659   0.0971 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 30.792  on 2057  degrees of freedom
## Residual deviance: 22.629  on 2054  degrees of freedom
## AIC: 183.04
##
## Number of Fisher Scoring iterations: 6
```

Doing this, the process stops very late, having only age0_20, PT_fact_big and PT_time_big left. The problem here comes from the above discussed problem of the overdispersion when using the binomial model.

The quasibinomial model on one side shows significances of parameters, but does not allow a stepwise variable selection with stepAIC function. The RMSE and R square values are on the other hand exactly the same, showing identic prediction behaviour. Therefore, the quasibinomial model can be used to selected parameters for a further try for the GLM. One possibility is still an option: The parameter selection can be done by taking the significant parameters from ta linear model approach, having the assumption that the same parameters are relevant for a linear model as for a GLM with the family binomial or quasibinomial. Therefore, a linear model is now built first before the analysis of the GLM can go further.

2.3.2 Linear Model (LM) (GA)

Now, a linear model is built, which models the absolute numbers of GA's per municipality. Due to the restriction of a linear model, it is not possible to model shares in a population with a linear model, as for a linear model, also negative values and values above 1 can occur which contradicts the possible range of the target variable. Therefore, the count dataset is used here. As is the case with the share dataset, all values except the target variable are normalized in the count data.

The main goal of the linear model is to gain insights into the automatic parameter selection when applying a stepwise AIC-selection approach. This can be used afterwards for the GLM again.

```
# First model with all predictors including defined interaction terms:
lm.count.01 <- lm(GA_BFS ~ . -HTA_BFS - fn_tck_BFS
                  + bus_count:bus_stat
                  + train_count:train_stat
                  + other_count:other_stat
                  + PT_fact_big:PT_fact_medium
                  + PT_time_big:PT_fact_big
                  + PT_time_medium:PT_time_big
                  + PT_fact_medium:PT_time_medium,
                data = d.norm.count.train)

summary(lm.count.01) ## looks generally good
```

```
##
## Call:
## lm(formula = GA_BFS ~ . - HTA_BFS - fn_tck_BFS + bus_count:bus_stat +
##     train_count:train_stat + other_count:other_stat + PT_fact_big:PT_fact_medium +
##     PT_time_big:PT_fact_big + PT_time_medium:PT_time_big + PT_fact_medium:PT_time_medium,
##     data = d.norm.count.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.41489 -0.25064  0.03564  0.28920  2.24066
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   6.935364   1.596205   4.345 1.48e-05 ***
## pop_count_BFS                 0.974372   0.882191   1.104  0.26955
## single_count_BFS              0.465120   0.432107   1.076  0.28191
## married_count_BFS            -0.122698   0.412669  -0.297  0.76625
## widowed_count_BFS            -0.072389   0.080456  -0.900  0.36840
## age0_20cnt                   -0.376662   0.121317  -3.105  0.00194 **
## age20_40cnt                  -0.392180   0.132593  -2.958  0.00314 **
## age40_60cnt                  -0.518610   0.182295  -2.845  0.00450 **
## birth_munic_cnt               0.091743   0.014283   6.423 1.75e-10 ***
## birth_cant_cnt                0.171636   0.033092   5.187 2.41e-07 ***
## birth_CH_cnt                  0.176513   0.031729   5.563 3.10e-08 ***
## male_cnt                     -0.976535   0.312777  -3.122  0.00183 **
## resid_0_1y_cnt                0.056169   0.044248   1.269  0.20448
## resid_1_5y_cnt               -0.030236   0.066738  -0.453  0.65057
## resid_6_10y_cnt               0.163556   0.053432   3.061  0.00224 **
## hh_1_cnt                      0.376674   0.083306   4.522 6.59e-06 ***
## hh_2_cnt                      0.633354   0.123804   5.116 3.50e-07 ***
## hh_3_5_cnt                    0.806233   0.181431   4.444 9.45e-06 ***
## PT_time_medium               -1.137606   0.268313  -4.240 2.36e-05 ***
## PT_time_big                  -1.382030   0.247884  -5.575 2.89e-08 ***
## bus_count                    -0.014350   0.007381  -1.944  0.05203 .
## other_count                   0.024251   0.011061   2.192  0.02849 *
## train_count                   0.015947   0.006299   2.532  0.01145 *
## bus_stat                      0.008816   0.079519   0.111  0.91174
## other_stat                   -0.317758   0.173837  -1.828  0.06775 .
```

```
## train_stat                    -0.288109   0.236471  -1.218  0.22326
## Combustion                    -0.657271   0.094394  -6.963 4.84e-12 ***
## Electric                       0.076963   0.034421   2.236  0.02549 *
## languagefr                    -0.367450   0.039966  -9.194  < 2e-16 ***
## languageit                    -1.177570   0.085592 -13.758  < 2e-16 ***
## languagerm                    -0.495890   0.083023  -5.973 2.86e-09 ***
## PT_fact_big                   -0.763953   0.454947  -1.679  0.09331 .
## PT_fact_medium                 0.517505   0.355054   1.458  0.14516
## bus_count:bus_stat             0.004921   0.005533   0.889  0.37394
## train_count:train_stat         0.024200   0.018400   1.315  0.18863
## other_count:other_stat         0.018868   0.017930   1.052  0.29282
## PT_fact_big:PT_fact_medium     0.285187   0.099452   2.868  0.00419 **
## PT_time_big:PT_fact_big       -0.034814   0.097689  -0.356  0.72160
## PT_time_medium:PT_time_big     0.307423   0.051634   5.954 3.21e-09 ***
## PT_time_medium:PT_fact_medium -0.227370   0.088454  -2.571  0.01024 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.444 on 1608 degrees of freedom
## Multiple R-squared:  0.8999, Adjusted R-squared:  0.8975
## F-statistic: 370.8 on 39 and 1608 DF,  p-value: < 2.2e-16
```

The output looks good in general, but to reduce the number of parameters, a stepwise selection model is applied:

```
lm.count.02 <- MASS::stepAIC(lm.count.01, direction = "both", trace = FALSE)
summary(lm.count.02)
```

```
##
## Call:
## lm(formula = GA_BFS ~ pop_count_BFS + single_count_BFS + age0_20cnt +
##     age20_40cnt + age40_60cnt + birth_munic_cnt + birth_cant_cnt +
##     birth_CH_cnt + male_cnt + resid_6_10y_cnt + hh_1_cnt + hh_2_cnt +
##     hh_3_5_cnt + PT_time_medium + PT_time_big + bus_count + other_count +
##     train_count + bus_stat + other_stat + train_stat + Combustion +
##     Electric + language + PT_fact_big + PT_fact_medium + train_count:train_stat +
##     PT_fact_big:PT_fact_medium + PT_time_medium:PT_time_big +
##     PT_time_medium:PT_fact_medium, data = d.norm.count.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.49979 -0.24782  0.03127  0.29266  2.21727
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   7.389653   1.249252   5.915 4.04e-09 ***
## pop_count_BFS                 0.597815   0.361864   1.652 0.098720 .
## single_count_BFS              0.625124   0.270533   2.311 0.020974 *
## age0_20cnt                   -0.378723   0.119115  -3.179 0.001503 **
## age20_40cnt                  -0.329670   0.122812  -2.684 0.007341 **
## age40_60cnt                  -0.445163   0.173835  -2.561 0.010532 *
## birth_munic_cnt               0.087921   0.013452   6.536 8.47e-11 ***
## birth_cant_cnt                0.162142   0.032024   5.063 4.60e-07 ***
```

```
## birth_CH_cnt                   0.172325   0.030621   5.628 2.15e-08 ***
## male_cnt                      -0.997199   0.310419  -3.212 0.001342 **
## resid_6_10y_cnt                0.171107   0.051604   3.316 0.000934 ***
## hh_1_cnt                       0.361177   0.071020   5.086 4.09e-07 ***
## hh_2_cnt                       0.637414   0.123072   5.179 2.51e-07 ***
## hh_3_5_cnt                     0.771965   0.176597   4.371 1.31e-05 ***
## PT_time_medium                -1.086170   0.260857  -4.164 3.29e-05 ***
## PT_time_big                   -1.408782   0.211993  -6.645 4.12e-11 ***
## bus_count                     -0.014090   0.007314  -1.927 0.054214 .
## other_count                    0.022056   0.010926   2.019 0.043678 *
## train_count                    0.015475   0.006252   2.475 0.013411 *
## bus_stat                       0.070275   0.027886   2.520 0.011828 *
## other_stat                    -0.141396   0.072318  -1.955 0.050734 .
## train_stat                    -0.333212   0.233182  -1.429 0.153203
## Combustion                    -0.673038   0.093263  -7.217 8.17e-13 ***
## Electric                       0.087520   0.033107   2.644 0.008283 **
## languagefr                    -0.363455   0.035783 -10.157  < 2e-16 ***
## languageit                    -1.172827   0.081623 -14.369  < 2e-16 ***
## languagerm                    -0.492835   0.081906  -6.017 2.19e-09 ***
## PT_fact_big                   -0.921344   0.172393  -5.344 1.04e-07 ***
## PT_fact_medium                 0.562345   0.349703   1.608 0.108017
## train_count:train_stat         0.028167   0.018054   1.560 0.118918
## PT_fact_big:PT_fact_medium     0.289348   0.098956   2.924 0.003504 **
## PT_time_medium:PT_time_big     0.299755   0.050573   5.927 3.76e-09 ***
## PT_time_medium:PT_fact_medium -0.240021   0.086699  -2.768 0.005697 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4437 on 1615 degrees of freedom
## Multiple R-squared:  0.8996, Adjusted R-squared:  0.8976
## F-statistic: 452.3 on 32 and 1615 DF,  p-value: < 2.2e-16
```

The stepAIC function gives me now a suggested formula, what can be used for a further try of the GLM. The coefficients show the weights.

2.3.3 Adapted GLM with family quasibinomial (GA)

Now, with a new parameter selection present, the process with the glm can continue now. As described further above, all parameters present after the stepwise parameter deletion with the function 'stepAIC' for the linear model, will be included in the glm-04 model:

```
# New run of quasibinomial model

glm.share.04 <- glm(GA_share ~ pop_count_BFS + single_share + age0_20 + age20_40 +
      age40_60 + birth_munic + birth_cant + birth_CH + male + resid_6_10y + hh_1 + hh_2 +
      hh_3_5 + PT_time_medium + PT_time_big + bus_stops_per_pop + other_stops_per_pop +
      train_stops_per_pop + bus_stat_per_1000 + other_stat_per_1000 + train_stat_per_1000 +
      comb_car_1000 + el_car_1000 + language + PT_fact_big + PT_fact_medium +
      train_stops_per_pop:train_stat_per_1000 +  PT_fact_big:PT_fact_medium +
      PT_time_medium:PT_time_big + PT_time_medium:PT_fact_medium,
                    family = quasibinomial, data = d.norm.share)

summary(glm.share.04)
```

```
##
## Call:
## glm(formula = GA_share ~ pop_count_BFS + single_share + age0_20 +
##     age20_40 + age40_60 + birth_munic + birth_cant + birth_CH +
##     male + resid_6_10y + hh_1 + hh_2 + hh_3_5 + PT_time_medium +
##     PT_time_big + bus_stops_per_pop + other_stops_per_pop + train_stops_per_pop +
##     bus_stat_per_1000 + other_stat_per_1000 + train_stat_per_1000 +
##     comb_car_1000 + el_car_1000 + language + PT_fact_big + PT_fact_medium +
##     train_stops_per_pop:train_stat_per_1000 + PT_fact_big:PT_fact_medium +
##     PT_time_medium:PT_time_big + PT_time_medium:PT_fact_medium,
##     family = quasibinomial, data = d.norm.share)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.39090  -0.06186  -0.01097   0.04640   0.47552
##
## Coefficients:
##                                          Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                             -3.095e+00  1.657e-02 -186.715  < 2e-16
## pop_count_BFS                           -5.133e-08  7.386e-07   -0.070 0.944595
## single_share                             2.594e-02  1.888e-02    1.374 0.169645
## age0_20                                 -1.017e-01  2.140e-02   -4.751 2.17e-06
## age20_40                                -3.962e-02  1.771e-02   -2.237 0.025374
## age40_60                                -1.180e-02  1.405e-02   -0.840 0.400989
## birth_munic                              1.246e-01  1.508e-02    8.264 2.51e-16
## birth_cant                               1.962e-01  2.560e-02    7.663 2.79e-14
## birth_CH                                 1.698e-01  2.400e-02    7.075 2.06e-12
## male                                    -5.899e-02  1.163e-02   -5.071 4.32e-07
## resid_6_10y                              2.432e-02  9.331e-03    2.606 0.009225
## hh_1                                     3.370e-01  7.574e-02    4.450 9.07e-06
## hh_2                                     1.940e-01  5.223e-02    3.715 0.000209
## hh_3_5                                   2.667e-01  7.253e-02    3.677 0.000242
## PT_time_medium                          -1.050e-01  1.171e-02   -8.965  < 2e-16
## PT_time_big                             -1.520e-01  1.976e-02   -7.693 2.24e-14
## bus_stops_per_pop                       -2.607e-02  1.252e-02   -2.083 0.037382
## other_stops_per_pop                     -1.824e-02  9.610e-03   -1.898 0.057827
## train_stops_per_pop                      8.684e-02  1.790e-02    4.852 1.32e-06
## bus_stat_per_1000                        1.257e-01  1.379e-02    9.116  < 2e-16
## other_stat_per_1000                      3.254e-02  1.130e-02    2.880 0.004022
## train_stat_per_1000                     -5.844e-02  2.866e-02   -2.039 0.041575
## comb_car_1000                           -8.673e-02  1.309e-02   -6.627 4.39e-11
## el_car_1000                              3.846e-02  1.226e-02    3.137 0.001731
## languagefr                              -3.068e-01  3.205e-02   -9.574  < 2e-16
## languageit                              -1.041e+00  8.885e-02  -11.719  < 2e-16
## languagerm                              -3.932e-01  8.561e-02   -4.593 4.64e-06
## PT_fact_big                             -1.672e-01  1.357e-02  -12.327  < 2e-16
## PT_fact_medium                           5.596e-03  1.218e-02    0.459 0.645978
## train_stops_per_pop:train_stat_per_1000  8.287e-04  2.743e-03    0.302 0.762604
## PT_fact_big:PT_fact_medium               3.707e-02  9.886e-03    3.750 0.000182
## PT_time_medium:PT_time_big               4.252e-02  6.491e-03    6.551 7.24e-11
## PT_time_medium:PT_fact_medium           -2.554e-02  1.208e-02   -2.115 0.034518
##
## (Intercept)                             ***
## pop_count_BFS
```

```
## single_share
## age0_20                                  ***
## age20_40                                 *
## age40_60
## birth_munic                              ***
## birth_cant                               ***
## birth_CH                                 ***
## male                                     ***
## resid_6_10y                              **
## hh_1                                     ***
## hh_2                                     ***
## hh_3_5                                   ***
## PT_time_medium                           ***
## PT_time_big                              ***
## bus_stops_per_pop                        *
## other_stops_per_pop                      .
## train_stops_per_pop                      ***
## bus_stat_per_1000                        ***
## other_stat_per_1000                      **
## train_stat_per_1000                      *
## comb_car_1000                            ***
## el_car_1000                              **
## languagefr                               ***
## languageit                               ***
## languagerm                               ***
## PT_fact_big                              ***
## PT_fact_medium
## train_stops_per_pop:train_stat_per_1000
## PT_fact_big:PT_fact_medium               ***
## PT_time_medium:PT_time_big               ***
## PT_time_medium:PT_fact_medium            *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasibinomial family taken to be 0.007750204)
##
##     Null deviance: 30.792  on 2057  degrees of freedom
## Residual deviance: 15.040  on 2025  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 7
```

Still, 33 parameters are too much here. In order to reduce further, the 10 lowest t values are removed. For the quasibinomial model, the coefficient t-value is a measure of how many standard deviations our coefficient estimate is far away from 0

```
# final run of glm with 10 parameters less present
glm.share.05 <- glm(GA_share ~  age0_20 + birth_munic + birth_cant + birth_CH + male +
    resid_6_10y + hh_1 + hh_2 + hh_3_5 + PT_time_medium + PT_time_big +
    train_stops_per_pop + bus_stat_per_1000 + other_stat_per_1000 + comb_car_1000 +
    el_car_1000 + language + PT_fact_big + PT_fact_big:PT_fact_medium +
    PT_time_medium:PT_time_big, family = quasibinomial, data = d.norm.share)
```

```
summary(glm.share.05)
```

```
##
## Call:
## glm(formula = GA_share ~ age0_20 + birth_munic + birth_cant +
##     birth_CH + male + resid_6_10y + hh_1 + hh_2 + hh_3_5 + PT_time_medium +
##     PT_time_big + train_stops_per_pop + bus_stat_per_1000 + other_stat_per_1000 +
##     comb_car_1000 + el_car_1000 + language + PT_fact_big + PT_fact_big:PT_fact_medium +
##     PT_time_medium:PT_time_big, family = quasibinomial, data = d.norm.share)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.42614  -0.06265  -0.01068   0.04589   0.48025
##
## Coefficients:
##                              Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                -3.099458   0.015537 -199.489  < 2e-16 ***
## age0_20                    -0.076546   0.018808   -4.070 4.88e-05 ***
## birth_munic                 0.126392   0.014051    8.995  < 2e-16 ***
## birth_cant                  0.212734   0.023702    8.975  < 2e-16 ***
## birth_CH                    0.182367   0.022724    8.025 1.69e-15 ***
## male                       -0.059117   0.010754   -5.497 4.35e-08 ***
## resid_6_10y                 0.025081   0.009400    2.668 0.007690 **
## hh_1                        0.327878   0.075230    4.358 1.38e-05 ***
## hh_2                        0.188176   0.051401    3.661 0.000258 ***
## hh_3_5                      0.245320   0.070513    3.479 0.000514 ***
## PT_time_medium             -0.100569   0.011424   -8.803  < 2e-16 ***
## PT_time_big                -0.159470   0.019136   -8.333  < 2e-16 ***
## train_stops_per_pop         0.043443   0.009567    4.541 5.93e-06 ***
## bus_stat_per_1000           0.114270   0.010718   10.661  < 2e-16 ***
## other_stat_per_1000         0.023328   0.008699    2.682 0.007385 **
## comb_car_1000              -0.089961   0.012574   -7.155 1.17e-12 ***
## el_car_1000                 0.040993   0.011848    3.460 0.000551 ***
## languagefr                 -0.302939   0.030302   -9.997  < 2e-16 ***
## languageit                 -0.971124   0.085872  -11.309  < 2e-16 ***
## languagerm                 -0.370305   0.085245   -4.344 1.47e-05 ***
## PT_fact_big                -0.164781   0.012082  -13.638  < 2e-16 ***
## PT_fact_big:PT_fact_medium  0.029749   0.009636    3.087 0.002046 **
## PT_time_medium:PT_time_big  0.044916   0.006281    7.151 1.20e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasibinomial family taken to be 0.007807145)
##
##     Null deviance: 30.792  on 2057  degrees of freedom
## Residual deviance: 15.254  on 2035  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 7
```

At least, all parameters left appear to be highly significant here. The model evaluation in a later section will show, how this model performs.

2.4 Modelling Half Fare tickets (HTA)

The same procedure shown for the GA is also valid for the HTA and will not be commented the same way here.

2.4.1 Generalized Linear model (GLM) with family "Binomial" (HTA)

The same procedure as used for the GA is applied here for the HTA. The first step with the selection of interactions is not done here, as the analysis from the GA should be enough for all target variables. The before defined interaction terms are taken as well into this model here:

The first run is done by using all possible influence factors and the already defined interaction terms:

```
#####  Modelling GLM for HTA with share data:

## Interactions: For interaction terms, the same selection is taken as described
## for the GA model above:

#################### Model set up 01 ######################
glm.HTA.share.01 <- glm(HTA_share ~ . - GA_share - FNT_share
                    + bus_stops_per_pop:bus_stat_per_1000
                    + train_stops_per_pop:train_stat_per_1000
                    + other_stops_per_pop:other_stops_per_pop
                    + PT_fact_big:PT_fact_medium
                    + PT_time_big:PT_fact_big
                    + PT_time_medium:PT_time_big
                    + PT_fact_medium:PT_time_medium,
                family = binomial, data=d.norm.share)
summary(glm.HTA.share.01)
```

```
Call:
glm(formula = HTA_share ~ . - GA_share - FNT_share + bus_stops_per_pop:bus_stat_per_1000 +
    train_stops_per_pop:train_stat_per_1000 + other_stops_per_pop:other_stops_per_pop +
    PT_fact_big:PT_fact_medium + PT_time_big:PT_fact_big + PT_time_medium:PT_time_big +
    PT_fact_medium:PT_time_medium, family = binomial, data = d.norm.share)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.58964  -0.07671  -0.00522   0.06892   0.57825

Coefficients:
                                          Estimate Std. Error z value Pr(>|z|)
(Intercept)                             -7.781e-01  8.907e-02  -8.736  < 2e-16 ***
languagefr                              -5.724e-01  1.833e-01  -3.122  0.00179 **
languageit                              -1.491e+00  5.103e-01  -2.923  0.00347 **
languagerm                              -1.753e-01  4.045e-01  -0.433  0.66479
pop_count_BFS                            6.017e-07  4.275e-06   0.141  0.88808
single_share                             1.526e-01  1.585e-01   0.963  0.33561
married_share                            7.800e-02  1.397e-01   0.558  0.57670
widowed_share                           -2.510e-02  1.066e-01  -0.235  0.81393
age0_20                                 -9.761e-02  1.201e-01  -0.813  0.41646
age20_40                                -1.468e-01  9.918e-02  -1.480  0.13895
age40_60                                -2.761e-02  7.740e-02  -0.357  0.72130
birth_munic                              4.534e-02  9.292e-02   0.488  0.62555
birth_cant                               3.974e-02  1.457e-01   0.273  0.78500
birth_CH                                 1.023e-02  1.328e-01   0.077  0.93862
male                                    -4.623e-02  6.285e-02  -0.736  0.46199
resid_0_1y                              -7.454e-03  6.822e-02  -0.109  0.91300
resid_1_5y                               3.322e-02  7.658e-02   0.434  0.66441
resid_6_10y                              5.067e-03  5.543e-02   0.091  0.92716
hh_1                                     4.595e-01  3.751e-01   1.225  0.22065
hh_2                                     3.087e-01  2.634e-01   1.172  0.24125
hh_3_5                                   4.088e-01  3.646e-01   1.121  0.26222
PT_time_medium                          -9.366e-02  6.099e-02  -1.536  0.12463
PT_time_big                             -3.793e-02  1.112e-01  -0.341  0.73303
PT_fact_big                             -8.454e-02  6.919e-02  -1.222  0.22175
PT_fact_medium                           4.066e-02  6.143e-02   0.662  0.50806
bus_stops_per_pop                       -6.242e-03  7.283e-02  -0.086  0.93171
train_stops_per_pop                      4.394e-02  9.468e-02   0.464  0.64259
other_stops_per_pop                      5.569e-03  6.141e-02   0.091  0.92774
bus_stat_per_1000                        4.028e-02  1.103e-01   0.365  0.71496
train_stat_per_1000                      1.692e-02  1.467e-01   0.115  0.90823
other_stat_per_1000                      3.073e-02  6.464e-02   0.475  0.63446
comb_car_1000                           -9.419e-02  6.519e-02  -1.445  0.14850
el_car_1000                              9.124e-02  6.323e-02   1.443  0.14903
inbound_share                           -2.212e-02  7.049e-02  -0.314  0.75371
outbound_share                           2.431e-02  7.708e-02   0.315  0.75249
bus_stops_per_pop:bus_stat_per_1000      9.582e-04  1.403e-02   0.068  0.94554
train_stops_per_pop:train_stat_per_1000 -3.280e-03  1.460e-02  -0.225  0.82229
PT_fact_big:PT_fact_medium               1.836e-02  4.894e-02   0.375  0.70755
PT_time_big:PT_fact_big                 -4.264e-02  8.160e-02  -0.523  0.60129
PT_time_medium:PT_time_big               2.095e-02  3.500e-02   0.599  0.54940
PT_time_medium:PT_fact_medium            5.114e-03  6.202e-02   0.082  0.93427
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 98.036  on 2057  degrees of freedom
Residual deviance: 28.578  on 2017  degrees of freedom
AIC: 1442.2

Number of Fisher Scoring iterations: 5
```

```
############### Stepwise parameter selection forward and backwards!############
glm.HTA.share.02 <- stepAIC(glm.HTA.share.01, direction="both", trace=FALSE)
summary(glm.HTA.share.02)
```

```
##
## Call:
## glm(formula = HTA_share ~ language + age20_40 + PT_fact_big +
##     comb_car_1000 + el_car_1000, family = binomial, data = d.norm.share)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.57252  -0.09774  -0.00596   0.08716   0.59063
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -0.75976    0.06050 -12.559  < 2e-16 ***
## languagefr    -0.57742    0.12413  -4.652 3.29e-06 ***
## languageit    -1.33328    0.29795  -4.475 7.65e-06 ***
## languagerm    -0.09659    0.33562  -0.288   0.7735
## age20_40      -0.09506    0.05286  -1.799   0.0721 .
## PT_fact_big   -0.10075    0.05687  -1.772   0.0764 .
## comb_car_1000 -0.11601    0.05681  -2.042   0.0411 *
## el_car_1000    0.10418    0.05345   1.949   0.0513 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 98.036  on 2057  degrees of freedom
## Residual deviance: 40.345  on 2050  degrees of freedom
## AIC: 1380
##
## Number of Fisher Scoring iterations: 5
```

Here, the stepwise parameter selection looks a bit better, giving still 5 significant parameters (including language with three manifestations) instead of 3 as it is the case for the GA. In order to achieve a consistent procedure, a linear model is nevertheless made again in order to transfer the variables relevant there into the next run of the GLM.

2.4.2 Linear Model (LM) (HTA)

Now the linear model will be identified:

```
# Model with all predictors including defined interaction terms
lm.count.HTA <- lm(HTA_BFS ~ . -GA_BFS - fn_tck_BFS
                   + bus_count:bus_stat
                   + train_count:train_stat
                   + other_count:other_stat
                   + PT_fact_big:PT_fact_medium
                   + PT_time_big:PT_fact_big
                   + PT_time_medium:PT_time_big
                   + PT_fact_medium:PT_time_medium,
```

```
                  data = d.norm.count.train)


### Stepwise selection of parameters:
lm.count.HTA2 <- MASS::stepAIC(lm.count.HTA, direction = "both", trace = FALSE)
summary(lm.count.HTA2)
```

```
##
## Call:
## lm(formula = HTA_BFS ~ single_count_BFS + married_count_BFS +
##     widowed_count_BFS + age0_20cnt + age20_40cnt + age40_60cnt +
##     birth_munic_cnt + birth_cant_cnt + male_cnt + resid_1_5y_cnt +
##     hh_1_cnt + hh_2_cnt + hh_3_5_cnt + PT_time_medium + PT_time_big +
##     bus_count + other_count + bus_stat + train_stat + Combustion +
##     Electric + language + PT_fact_big + PT_fact_medium + PT_fact_big:PT_fact_medium +
##     PT_time_big:PT_fact_big + PT_time_medium:PT_time_big, data = d.norm.count.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.31512 -0.12024  0.01251  0.13003  1.11071
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  1.909632   0.569091   3.356  0.00081 ***
## single_count_BFS             0.959184   0.094025  10.201  < 2e-16 ***
## married_count_BFS            0.501435   0.088806   5.646 1.93e-08 ***
## widowed_count_BFS           -0.067731   0.030561  -2.216  0.02681 *
## age0_20cnt                  -0.329199   0.056057  -5.873 5.20e-09 ***
## age20_40cnt                 -0.562584   0.061035  -9.217  < 2e-16 ***
## age40_60cnt                 -0.132514   0.085851  -1.544  0.12290
## birth_munic_cnt              0.010673   0.006452   1.654  0.09827 .
## birth_cant_cnt               0.036827   0.011494   3.204  0.00138 **
## male_cnt                    -0.376998   0.147149  -2.562  0.01050 *
## resid_1_5y_cnt               0.058926   0.030368   1.940  0.05250 .
## hh_1_cnt                     0.104446   0.037930   2.754  0.00596 **
## hh_2_cnt                     0.566852   0.056548  10.024  < 2e-16 ***
## hh_3_5_cnt                   0.529794   0.086027   6.158 9.25e-10 ***
## PT_time_medium              -0.278742   0.105355  -2.646  0.00823 **
## PT_time_big                  0.035386   0.116022   0.305  0.76041
## bus_count                   -0.011167   0.003470  -3.219  0.00131 **
## other_count                  0.017394   0.002112   8.237 3.61e-16 ***
## bus_stat                     0.034386   0.013360   2.574  0.01015 *
## train_stat                   0.042100   0.012841   3.278  0.00107 **
## Combustion                  -0.453069   0.043979 -10.302  < 2e-16 ***
## Electric                     0.129083   0.016090   8.023 1.97e-15 ***
## languagefr                  -0.400267   0.017720 -22.588  < 2e-16 ***
## languageit                  -1.035626   0.038597 -26.832  < 2e-16 ***
## languagerm                  -0.067453   0.038989  -1.730  0.08381 .
## PT_fact_big                  0.364195   0.205748   1.770  0.07690 .
## PT_fact_medium              -0.125112   0.076773  -1.630  0.10338
## PT_fact_big:PT_fact_medium   0.138148   0.045125   3.061  0.00224 **
## PT_time_big:PT_fact_big     -0.184802   0.044445  -4.158 3.38e-05 ***
## PT_time_medium:PT_time_big   0.039938   0.024178   1.652  0.09877 .
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2116 on 1618 degrees of freedom
## Multiple R-squared:  0.975,  Adjusted R-squared:  0.9746
## F-statistic:  2178 on 29 and 1618 DF,  p-value: < 2.2e-16
```

```
summary(lm.count.HTA)
```

```
##
## Call:
## lm(formula = HTA_BFS ~ . - GA_BFS - fn_tck_BFS + bus_count:bus_stat +
##     train_count:train_stat + other_count:other_stat + PT_fact_big:PT_fact_medium +
##     PT_time_big:PT_fact_big + PT_time_medium:PT_time_big + PT_fact_medium:PT_time_medium,
##     data = d.norm.count.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.31066 -0.11801  0.01298  0.13069  1.08237
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   2.161818   0.761934   2.837 0.004607 **
## pop_count_BFS                -0.414047   0.421105  -0.983 0.325638
## single_count_BFS              1.154981   0.206262   5.600 2.52e-08 ***
## married_count_BFS             0.671754   0.196984   3.410 0.000665 ***
## widowed_count_BFS            -0.048339   0.038405  -1.259 0.208329
## age0_20cnt                   -0.350760   0.057910  -6.057 1.72e-09 ***
## age20_40cnt                  -0.551971   0.063292  -8.721  < 2e-16 ***
## age40_60cnt                  -0.125115   0.087017  -1.438 0.150677
## birth_munic_cnt               0.012419   0.006818   1.821 0.068721 .
## birth_cant_cnt                0.043698   0.015796   2.766 0.005733 **
## birth_CH_cnt                  0.013782   0.015146   0.910 0.362963
## male_cnt                     -0.389611   0.149301  -2.610 0.009150 **
## resid_0_1y_cnt               -0.014630   0.021122  -0.693 0.488629
## resid_1_5y_cnt                0.064947   0.031857   2.039 0.041640 *
## resid_6_10y_cnt               0.029652   0.025505   1.163 0.245177
## hh_1_cnt                      0.117742   0.039765   2.961 0.003112 **
## hh_2_cnt                      0.564346   0.059097   9.550  < 2e-16 ***
## hh_3_5_cnt                    0.524530   0.086604   6.057 1.73e-09 ***
## PT_time_medium               -0.238336   0.128077  -1.861 0.062942 .
## PT_time_big                   0.027718   0.118325   0.234 0.814818
## bus_count                    -0.011591   0.003523  -3.290 0.001023 **
## other_count                   0.019034   0.005280   3.605 0.000322 ***
## train_count                  -0.001481   0.003007  -0.493 0.622309
## bus_stat                      0.055386   0.037958   1.459 0.144722
## other_stat                    0.016396   0.082979   0.198 0.843393
## train_stat                    0.092564   0.112877   0.820 0.412315
## Combustion                   -0.456099   0.045058 -10.122  < 2e-16 ***
## Electric                      0.129709   0.016430   7.894 5.35e-15 ***
## languagefr                   -0.392746   0.019077 -20.587  < 2e-16 ***
## languageit                   -1.026481   0.040856 -25.124  < 2e-16 ***
## languagerm                   -0.070804   0.039630  -1.787 0.074187 .
## PT_fact_big                   0.376991   0.217165   1.736 0.082761 .
```

```
## PT_fact_medium                -0.002438   0.169482  -0.014 0.988525
## bus_count:bus_stat            -0.001520   0.002641  -0.575 0.565065
## train_count:train_stat        -0.003458   0.008783  -0.394 0.693853
## other_count:other_stat        -0.003252   0.008559  -0.380 0.704048
## PT_fact_big:PT_fact_medium     0.152868   0.047472   3.220 0.001307 **
## PT_time_big:PT_fact_big       -0.193392   0.046631  -4.147 3.54e-05 ***
## PT_time_medium:PT_time_big     0.043471   0.024647   1.764 0.077970 .
## PT_time_medium:PT_fact_medium -0.035759   0.042223  -0.847 0.397173
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2119 on 1608 degrees of freedom
## Multiple R-squared:  0.9751, Adjusted R-squared:  0.9745
## F-statistic:  1614 on 39 and 1608 DF,  p-value: < 2.2e-16
```

The stepAIC function gives me now a suggested formula, what can be used for a further try of the GLM. The coefficients show the weights.

2.4.3 Adapted GLM with family quasibinomial (HTA)

Also here, all parameters present after the stepwise parameter deletion with the function 'stepAIC' for the linear model, will be included in the next GLM model:

Let's have a look at the next model run:

```
# New run of quasibinomial model

glm.HTA.share.03 <- glm(HTA_share ~ single_share + married_share + widowed_share +
      age0_20 + age20_40 + age40_60 + birth_munic + birth_cant + male + resid_1_5y +
      hh_1 + hh_2 + hh_3_5 + PT_time_medium + PT_time_big + bus_stops_per_pop +
      other_stops_per_pop + bus_stat_per_1000 + train_stat_per_1000 + comb_car_1000 +
      el_car_1000 + language + PT_fact_big + PT_fact_medium + PT_fact_big:PT_fact_medium +
      PT_time_big:PT_fact_big + PT_time_medium:PT_time_big,
      family = quasibinomial, data=d.norm.share)

summary(glm.HTA.share.03)
```

```
##
## Call:
## glm(formula = HTA_share ~ single_share + married_share + widowed_share +
##     age0_20 + age20_40 + age40_60 + birth_munic + birth_cant +
##     male + resid_1_5y + hh_1 + hh_2 + hh_3_5 + PT_time_medium +
##     PT_time_big + bus_stops_per_pop + other_stops_per_pop + bus_stat_per_1000 +
##     train_stat_per_1000 + comb_car_1000 + el_car_1000 + language +
##     PT_fact_big + PT_fact_medium + PT_fact_big:PT_fact_medium +
##     PT_time_big:PT_fact_big + PT_time_medium:PT_time_big, family = quasibinomial,
##     data = d.norm.share)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.60783  -0.07742  -0.00520   0.06695   0.58016
##
```

```
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -0.777410   0.009976 -77.930  < 2e-16 ***
## single_share                0.155443   0.018526   8.391  < 2e-16 ***
## married_share               0.077070   0.016495   4.672 3.17e-06 ***
## widowed_share              -0.030970   0.012505  -2.477 0.013346 *
## age0_20                    -0.101718   0.014152  -7.188 9.23e-13 ***
## age20_40                   -0.157035   0.011403 -13.771  < 2e-16 ***
## age40_60                   -0.026701   0.009225  -2.894 0.003840 **
## birth_munic                 0.035909   0.008016   4.480 7.89e-06 ***
## birth_cant                  0.030798   0.007590   4.058 5.15e-05 ***
## male                       -0.048081   0.007469  -6.438 1.51e-10 ***
## resid_1_5y                  0.028209   0.008668   3.255 0.001155 **
## hh_1                        0.460956   0.044837  10.281  < 2e-16 ***
## hh_2                        0.315471   0.030755  10.257  < 2e-16 ***
## hh_3_5                      0.412828   0.043206   9.555  < 2e-16 ***
## PT_time_medium             -0.093181   0.007222 -12.902  < 2e-16 ***
## PT_time_big                -0.033584   0.012043  -2.789 0.005340 **
## bus_stops_per_pop          -0.011185   0.008502  -1.315 0.188494
## other_stops_per_pop         0.027215   0.005865   4.641 3.70e-06 ***
## bus_stat_per_1000           0.044728   0.010572   4.231 2.43e-05 ***
## train_stat_per_1000         0.026425   0.006343   4.166 3.23e-05 ***
## comb_car_1000              -0.096891   0.007569 -12.800  < 2e-16 ***
## el_car_1000                 0.087232   0.007389  11.805  < 2e-16 ***
## languagefr                 -0.565877   0.020978 -26.974  < 2e-16 ***
## languageit                 -1.513014   0.058383 -25.915  < 2e-16 ***
## languagerm                 -0.179718   0.048199  -3.729 0.000198 ***
## PT_fact_big                -0.088425   0.008117 -10.894  < 2e-16 ***
## PT_fact_medium              0.037766   0.007212   5.237 1.80e-07 ***
## PT_fact_big:PT_fact_medium  0.018738   0.005709   3.282 0.001047 **
## PT_time_big:PT_fact_big    -0.044536   0.009448  -4.714 2.60e-06 ***
## PT_time_medium:PT_time_big  0.020038   0.004117   4.867 1.22e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasibinomial family taken to be 0.01448846)
##
##     Null deviance: 98.036  on 2057  degrees of freedom
## Residual deviance: 29.329  on 2028  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

still, 30 parameters are too much here. IN order to reduce further, the 6 lowest t values are removed. => here to get all parameters with high t value, marked as *** The coefficient t-value is a measure of how many standard deviations our coefficient estimate is far away from 0

```
glm.HTA.share.04 <- glm(HTA_share ~ single_share + married_share + age0_20 +
                        age20_40 + birth_munic + birth_cant + male + hh_1 + hh_2 +
                        hh_3_5 + PT_time_medium + other_stops_per_pop +
                        bus_stat_per_1000 + train_stat_per_1000 + comb_car_1000 +
                        el_car_1000 + language + PT_fact_big + PT_fact_medium +
                        PT_time_big:PT_fact_big + PT_time_medium:PT_time_big,
                    family = quasibinomial, data=d.norm.share)
```

```
summary(glm.HTA.share.04)
```

```
##
## Call:
## glm(formula = HTA_share ~ single_share + married_share + age0_20 +
##     age20_40 + birth_munic + birth_cant + male + hh_1 + hh_2 +
##     hh_3_5 + PT_time_medium + other_stops_per_pop + bus_stat_per_1000 +
##     train_stat_per_1000 + comb_car_1000 + el_car_1000 + language +
##     PT_fact_big + PT_fact_medium + PT_time_big:PT_fact_big +
##     PT_time_medium:PT_time_big, family = quasibinomial, data = d.norm.share)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.63082  -0.07697  -0.00346   0.06843   0.64535
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -0.768876   0.008840 -86.981  < 2e-16 ***
## single_share                0.161107   0.016373   9.840  < 2e-16 ***
## married_share               0.082751   0.014655   5.647 1.86e-08 ***
## age0_20                    -0.078824   0.013112  -6.011 2.17e-09 ***
## age20_40                   -0.127007   0.009235 -13.753  < 2e-16 ***
## birth_munic                 0.019953   0.007184   2.777 0.005532 **
## birth_cant                  0.024491   0.007235   3.385 0.000725 ***
## male                       -0.048770   0.007141  -6.829 1.12e-11 ***
## hh_1                        0.464048   0.044626  10.399  < 2e-16 ***
## hh_2                        0.327810   0.030394  10.785  < 2e-16 ***
## hh_3_5                      0.408575   0.042244   9.672  < 2e-16 ***
## PT_time_medium             -0.099748   0.007189 -13.875  < 2e-16 ***
## other_stops_per_pop         0.024943   0.005830   4.279 1.97e-05 ***
## bus_stat_per_1000           0.033598   0.008752   3.839 0.000127 ***
## train_stat_per_1000         0.025985   0.006207   4.186 2.96e-05 ***
## comb_car_1000              -0.105916   0.007430 -14.255  < 2e-16 ***
## el_car_1000                 0.099659   0.007164  13.911  < 2e-16 ***
## languagefr                 -0.544974   0.019877 -27.417  < 2e-16 ***
## languageit                 -1.592514   0.049827 -31.961  < 2e-16 ***
## languagerm                 -0.227308   0.043953  -5.172 2.55e-07 ***
## PT_fact_big                -0.090835   0.008148 -11.148  < 2e-16 ***
## PT_fact_medium              0.044095   0.007122   6.192 7.19e-10 ***
## PT_fact_big:PT_time_big    -0.043322   0.009482  -4.569 5.20e-06 ***
## PT_time_medium:PT_time_big  0.019171   0.003992   4.802 1.68e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasibinomial family taken to be 0.01483236)
##
##     Null deviance: 98.036  on 2057  degrees of freedom
## Residual deviance: 30.050  on 2034  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

At least, all parameters left appear to be highly significant here. The model evaluation in a later section will

show how this model performs.

2.5 Modelling Fare Network Tickets (FNT)

The same procedure shown for the GA and HTA is also valid for the FNT and will not be commented the same way here.

2.5.1 Generalized Linear Model (GLM) with family "Binomial" (FNT)

The same procedure as used for the GA is applied here for the FNT. The first step with the selection of interactions is not done here, as the analysis from the GA should be enough for all target variables. The earlier defined interaction terms are taken as well into this model here:

The first run is done by using all possible influence factors and the already defined interaction terms:

```
#####  Modelling GLM for FNT with share data:

## Interactions: For interaction terms, the same selection is taken as described
## for the GA model above:

################### Model set up 01 ######################
glm.FNT.share.01 <- glm(FNT_share ~ . - GA_share - HTA_share
                    + bus_stops_per_pop:bus_stat_per_1000
                    + train_stops_per_pop:train_stat_per_1000
                    + other_stops_per_pop:other_stops_per_pop
                    + PT_fact_big:PT_fact_medium
                    + PT_time_big:PT_fact_big
                    + PT_time_medium:PT_time_big
                    + PT_fact_medium:PT_time_medium,
                family = binomial, data=d.norm.share)
summary(glm.FNT.share.01)
```

```
Call:
glm(formula = FNT_share ~ . - GA_share - HTA_share + bus_stops_per_pop:bus_stat_per_1000 +
    train_stops_per_pop:train_stat_per_1000 + other_stops_per_pop:other_stops_per_pop +
    PT_fact_big:PT_fact_medium + PT_time_big:PT_fact_big + PT_time_medium:PT_time_big +
    PT_fact_medium:PT_time_medium, family = binomial, data = d.norm.share)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.49732  -0.14339  -0.03728   0.08088   0.57255

Coefficients:
                                          Estimate Std. Error z value Pr(>|z|)
(Intercept)                             -3.503e+00  2.259e-01 -15.512   <2e-16 ***
languagefr                               2.680e-01  3.969e-01   0.675    0.500
languageit                               1.421e+00  1.121e+00   1.267    0.205
languagerm                              -2.245e+00  3.568e+00  -0.629    0.529
pop_count_BFS                            4.792e-06  6.018e-06   0.796    0.426
single_share                             1.317e-02  3.604e-01   0.037    0.971
married_share                           -1.852e-01  3.062e-01  -0.605    0.545
widowed_share                           -1.428e-01  2.353e-01  -0.607    0.544
age0_20                                 -5.401e-03  2.473e-01  -0.022    0.983
age20_40                                -2.185e-01  2.125e-01  -1.028    0.304
age40_60                                -4.094e-02  1.679e-01  -0.244    0.807
birth_munic                             -2.611e-01  2.253e-01  -1.159    0.246
birth_cant                              -1.518e-01  2.953e-01  -0.514    0.607
birth_CH                                -2.863e-01  2.837e-01  -1.009    0.313
male                                    -6.135e-02  1.410e-01  -0.435    0.663
resid_0_1y                              -2.690e-02  1.378e-01  -0.195    0.845
resid_1_5y                              -1.925e-02  1.549e-01  -0.124    0.901
resid_6_10y                             -1.517e-02  1.495e-01  -0.101    0.919
hh_1                                    -1.401e-01  7.063e-01  -0.198    0.843
hh_2                                    -4.980e-02  4.927e-01  -0.101    0.919
hh_3_5                                  -5.103e-02  7.147e-01  -0.071    0.943
PT_time_medium                          -1.152e-01  1.394e-01  -0.827    0.409
PT_time_big                             -2.718e-01  2.576e-01  -1.055    0.291
PT_fact_big                             -2.205e-02  1.633e-01  -0.135    0.893
PT_fact_medium                           1.282e-01  1.342e-01   0.955    0.340
bus_stops_per_pop                        1.505e-01  1.484e-01   1.015    0.310
train_stops_per_pop                      5.646e-02  2.363e-01   0.239    0.811
other_stops_per_pop                     -9.806e-02  4.702e-01  -0.209    0.835
bus_stat_per_1000                       -1.368e-03  2.687e-01  -0.005    0.996
train_stat_per_1000                     -3.257e-02  3.941e-01  -0.083    0.934
other_stat_per_1000                      7.410e-03  1.703e-01   0.044    0.965
comb_car_1000                           -6.052e-02  1.434e-01  -0.422    0.673
el_car_1000                              9.216e-02  1.319e-01   0.699    0.485
inbound_share                           -5.983e-03  1.485e-01  -0.040    0.968
outbound_share                          -2.673e-02  1.897e-01  -0.141    0.888
bus_stops_per_pop:bus_stat_per_1000     -5.573e-03  3.144e-02  -0.177    0.859
train_stops_per_pop:train_stat_per_1000 -2.581e-03  4.549e-02  -0.057    0.955
PT_fact_big:PT_fact_medium              -8.752e-03  9.381e-02  -0.093    0.926
PT_time_big:PT_fact_big                  2.741e-02  1.901e-01   0.144    0.885
PT_time_medium:PT_time_big               4.671e-02  6.702e-02   0.697    0.486
PT_time_medium:PT_fact_medium            1.135e-01  1.290e-01   0.880    0.379
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 92.197  on 2057  degrees of freedom
Residual deviance: 51.104  on 2017  degrees of freedom
AIC: 263.99

Number of Fisher Scoring iterations: 8
```

```
############### Stepwise parameter selection forward and backwards!############
glm.FNT.share.02 <- stepAIC(glm.FNT.share.01, direction="both", trace=FALSE)
summary(glm.FNT.share.02)
```

```
##
## Call:
## glm(formula = FNT_share ~ married_share + age20_40 + birth_munic +
##     birth_cant + birth_CH + hh_3_5, family = binomial, data = d.norm.share)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.45127  -0.16718  -0.04545   0.09600   0.61640
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -3.2556     0.1229 -26.494  < 2e-16 ***
## married_share  -0.2654     0.1243  -2.134  0.03282 *
## age20_40       -0.2474     0.1231  -2.010  0.04447 *
## birth_munic    -0.3640     0.1579  -2.304  0.02121 *
## birth_cant     -0.3967     0.1713  -2.316  0.02055 *
## birth_CH       -0.5639     0.1820  -3.099  0.00194 **
## hh_3_5          0.1900     0.1015   1.872  0.06116 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 92.197  on 2057  degrees of freedom
## Residual deviance: 63.130  on 2051  degrees of freedom
## AIC: 195.38
##
## Number of Fisher Scoring iterations: 6
```

Here, the stepwise parameter selection gives back 6 significant parameters. In order to achieve a consistent procedure, a linear model is nevertheless made again in order to transfer the variables relevant there into the next run of the GLM.

2.5.2 Linear Model (LM) (FNT)

Now the linear model will be identified:

```
# Model with all predictors including defined interaction terms
lm.count.FNT <- lm(fn_tck_BFS ~ . -GA_BFS - HTA_BFS
                   + bus_count:bus_stat
                   + train_count:train_stat
                   + other_count:other_stat
                   + PT_fact_big:PT_fact_medium
                   + PT_time_big:PT_fact_big
                   + PT_time_medium:PT_time_big
                   + PT_fact_medium:PT_time_medium,
                  data = d.norm.count.train)
```

```r
lm.count.FNT2 <- MASS::stepAIC(lm.count.FNT, direction = "both", trace = FALSE)
summary(lm.count.FNT2)
```

```
##
## Call:
## lm(formula = fn_tck_BFS ~ single_count_BFS + married_count_BFS +
##     age20_40cnt + age40_60cnt + birth_munic_cnt + birth_cant_cnt +
##     birth_CH_cnt + male_cnt + resid_1_5y_cnt + hh_1_cnt + PT_time_medium +
##     PT_time_big + bus_count + other_count + train_count + bus_stat +
##     other_stat + train_stat + Combustion + Electric + language +
##     PT_fact_big + PT_fact_medium + bus_count:bus_stat + train_count:train_stat +
##     other_count:other_stat + PT_time_big:PT_fact_big + PT_time_medium:PT_time_big +
##     PT_time_medium:PT_fact_medium, data = d.norm.count.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6400 -0.6441  0.1699  0.8092  2.6282
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   19.679425   3.114060   6.320 3.38e-10 ***
## single_count_BFS               2.516410   0.437647   5.750 1.07e-08 ***
## married_count_BFS             -1.630979   0.433251  -3.765 0.000173 ***
## age20_40cnt                   -1.612113   0.293202  -5.498 4.45e-08 ***
## age40_60cnt                   -0.820709   0.408271  -2.010 0.044575 *
## birth_munic_cnt               -0.246423   0.035751  -6.893 7.82e-12 ***
## birth_cant_cnt                 0.230309   0.082383   2.796 0.005242 **
## birth_CH_cnt                  -0.192380   0.079059  -2.433 0.015066 *
## male_cnt                       1.658677   0.752646   2.204 0.027679 *
## resid_1_5y_cnt                 0.396994   0.161273   2.462 0.013935 *
## hh_1_cnt                      -0.219162   0.137817  -1.590 0.111975
## PT_time_medium                -4.237189   0.670591  -6.319 3.41e-10 ***
## PT_time_big                   -4.423755   0.636492  -6.950 5.27e-12 ***
## bus_count                      0.015476   0.019059   0.812 0.416891
## other_count                   -0.021848   0.028539  -0.766 0.444051
## train_count                    0.005934   0.016333   0.363 0.716433
## bus_stat                      -0.505571   0.205689  -2.458 0.014078 *
## other_stat                     0.766786   0.449621   1.705 0.088311 .
## train_stat                    -0.946986   0.613927  -1.543 0.123146
## Combustion                     0.465439   0.240100   1.939 0.052734 .
## Electric                       0.387637   0.085822   4.517 6.74e-06 ***
## languagefr                    -0.046085   0.089294  -0.516 0.605855
## languageit                     1.561043   0.215131   7.256 6.15e-13 ***
## languagerm                    -1.495839   0.212774  -7.030 3.03e-12 ***
## PT_fact_big                   -3.266925   1.081583  -3.021 0.002563 **
## PT_fact_medium                -1.926268   0.906598  -2.125 0.033761 *
## bus_count:bus_stat             0.037508   0.014299   2.623 0.008796 **
## train_count:train_stat         0.097347   0.047771   2.038 0.041733 *
## other_count:other_stat        -0.079439   0.046391  -1.712 0.087019 .
## PT_time_big:PT_fact_big        0.677385   0.250065   2.709 0.006823 **
## PT_time_medium:PT_time_big     0.709501   0.129642   5.473 5.13e-08 ***
## PT_time_medium:PT_fact_medium  0.570787   0.219871   2.596 0.009517 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.155 on 1616 degrees of freedom
## Multiple R-squared:  0.6438, Adjusted R-squared:  0.637
## F-statistic: 94.23 on 31 and 1616 DF,  p-value: < 2.2e-16
```

The stepAIC function gives us now a suggested formula, what can be used for a further try of the GLM. The coefficients show the weights of the parameters.

2.5.3 Adapted GLM with family quasibinomial (FNT)

Also here, all parameters present after the stepwise parameter deletion with the function 'stepAIC' for the linear model, will be included in the next GLM model:

```
# New run of quasibinomial model

glm.FNT.share.03 <- glm(FNT_share ~ single_share + married_share +
                          age20_40 + age40_60 + birth_munic + birth_cant +
                          birth_CH + male + resid_1_5y + hh_1 + PT_time_medium +
                          PT_time_big + bus_stops_per_pop + other_stops_per_pop +
                          train_stops_per_pop + bus_stat_per_1000 +
                          other_stat_per_1000 + train_stat_per_1000 +
                          comb_car_1000 + el_car_1000 + language + PT_fact_big +
                          PT_fact_medium + bus_stops_per_pop:bus_stat_per_1000 +
                          train_stops_per_pop:train_stat_per_1000 +
                          other_stops_per_pop:other_stat_per_1000 +
                          PT_time_big:PT_fact_big + PT_time_medium:PT_time_big +
                          PT_time_medium:PT_fact_medium,
                family = quasibinomial, data=d.norm.share)

summary(glm.FNT.share.03)
```

```
Call:
glm(formula = FNT_share ~ single_share + married_share + age20_40 +
    age40_60 + birth_munic + birth_cant + birth_CH + male + resid_1_5y +
    hh_1 + PT_time_medium + PT_time_big + bus_stops_per_pop +
    other_stops_per_pop + train_stops_per_pop + bus_stat_per_1000 +
    other_stat_per_1000 + train_stat_per_1000 + comb_car_1000 +
    el_car_1000 + language + PT_fact_big + PT_fact_medium +
bus_stops_per_pop:bus_stat_per_1000 +
    train_stops_per_pop:train_stat_per_1000 + other_stops_per_pop:other_stat_per_1000 +
    PT_time_big:PT_fact_big + PT_time_medium:PT_time_big + PT_time_medium:PT_fact_medium,
    family = quasibinomial, data = d.norm.share)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.50229  -0.14390  -0.03697   0.08263   0.58363

Coefficients:
                                          Estimate Std. Error  t value Pr(>|t|)
(Intercept)                              -3.478043   0.034221 -101.633  < 2e-16 ***
single_share                              0.139017   0.041944    3.314 0.000935 ***
married_share                            -0.101448   0.041667   -2.435 0.014988 *
age20_40                                 -0.194572   0.029686   -6.554 7.07e-11 ***
age40_60                                 -0.015517   0.023517   -0.660 0.509439
birth_munic                              -0.231389   0.030740   -7.527 7.75e-14 ***
birth_cant                               -0.185249   0.040347   -4.591 4.67e-06 ***
birth_CH                                 -0.303236   0.040404   -7.505 9.13e-14 ***
male                                     -0.051868   0.022223   -2.334 0.019693 *
resid_1_5y                               -0.021653   0.023664   -0.915 0.360281
hh_1                                     -0.084415   0.031848   -2.651 0.008098 **
PT_time_medium                           -0.137130   0.021677   -6.326 3.08e-10 ***
PT_time_big                              -0.298950   0.039461   -7.576 5.39e-14 ***
bus_stops_per_pop                         0.140268   0.022770    6.160 8.74e-10 ***
other_stops_per_pop                      -0.099662   0.109559   -0.910 0.363111
train_stops_per_pop                       0.038564   0.036944    1.044 0.296679
bus_stat_per_1000                         0.018558   0.040392    0.459 0.645970
other_stat_per_1000                       0.016137   0.028137    0.574 0.566354
train_stat_per_1000                      -0.022547   0.062274   -0.362 0.717346
comb_car_1000                            -0.067904   0.022523   -3.015 0.002602 **
el_car_1000                               0.100034   0.020393    4.905 1.01e-06 ***
languagefr                                0.259459   0.057207    4.535 6.08e-06 ***
languageit                                1.397463   0.172115    8.119 8.05e-16 ***
languagerm                               -2.207413   0.572322   -3.857 0.000118 ***
PT_fact_big                              -0.013506   0.025592   -0.528 0.597744
PT_fact_medium                            0.130882   0.020439    6.404 1.88e-10 ***
bus_stops_per_pop:bus_stat_per_1000      -0.005018   0.004969   -1.010 0.312668
train_stops_per_pop:train_stat_per_1000 -0.002977   0.007298   -0.408 0.683416
other_stops_per_pop:other_stat_per_1000 -0.003828   0.017364   -0.220 0.825545
PT_time_big:PT_fact_big                   0.022326   0.029563    0.755 0.450229
PT_time_medium:PT_time_big                0.051721   0.010269    5.037 5.16e-07 ***
PT_time_medium:PT_fact_medium             0.115926   0.019523    5.938 3.39e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for quasibinomial family taken to be 0.02578729)

    Null deviance: 92.197  on 2057  degrees of freedom
Residual deviance: 52.281  on 2026  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 9
```

Still, 32 parameters are too much here. In order to reduce further, about 10 - 15 parameters with the lowest t values should be removed. The coefficient t-value is a measure of how many standard deviations

our coefficient estimate is far away from 0. This will be used as the criterion to eliminate parameters from the model.

Now the final model with only selected parameters:

```
glm.FNT.share.04 <- glm(FNT_share ~ single_share + married_share +
                          age20_40 + birth_munic + birth_cant +
                          birth_CH + male + hh_1 + PT_time_medium +
                          PT_time_big + bus_stops_per_pop +
                          comb_car_1000 + el_car_1000 + language +
                          PT_fact_medium + PT_time_medium:PT_time_big +
                          PT_time_medium:PT_fact_medium,
                family = quasibinomial, data=d.norm.share)

summary(glm.FNT.share.04)
```

```
##
## Call:
## glm(formula = FNT_share ~ single_share + married_share + age20_40 +
##     birth_munic + birth_cant + birth_CH + male + hh_1 + PT_time_medium +
##     PT_time_big + bus_stops_per_pop + comb_car_1000 + el_car_1000 +
##     language + PT_fact_medium + PT_time_medium:PT_time_big +
##     PT_time_medium:PT_fact_medium, family = quasibinomial, data = d.norm.share)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.50178  -0.14437  -0.03755   0.08272   0.58142
##
## Coefficients:
##                                Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                   -3.484747   0.032447 -107.399  < 2e-16 ***
## single_share                   0.130257   0.041502    3.139 0.001722 **
## married_share                 -0.103897   0.041123   -2.527 0.011595 *
## age20_40                      -0.189211   0.025547   -7.406 1.89e-13 ***
## birth_munic                   -0.214817   0.027843   -7.715 1.87e-14 ***
## birth_cant                    -0.168764   0.036214   -4.660 3.36e-06 ***
## birth_CH                      -0.291671   0.038001   -7.675 2.54e-14 ***
## male                          -0.053364   0.019979   -2.671 0.007623 **
## hh_1                          -0.080547   0.030481   -2.643 0.008292 **
## PT_time_medium                -0.139599   0.021375   -6.531 8.22e-11 ***
## PT_time_big                   -0.303469   0.037653   -8.060 1.29e-15 ***
## bus_stops_per_pop              0.130462   0.017896    7.290 4.42e-13 ***
## comb_car_1000                 -0.071031   0.022331   -3.181 0.001491 **
## el_car_1000                    0.095915   0.020276    4.730 2.39e-06 ***
## languagefr                     0.260900   0.054959    4.747 2.21e-06 ***
## languageit                     1.381175   0.138398    9.980  < 2e-16 ***
## languagerm                    -2.187709   0.570398   -3.835 0.000129 ***
## PT_fact_medium                 0.121513   0.018626    6.524 8.61e-11 ***
## PT_time_medium:PT_time_big     0.053436   0.009791    5.458 5.41e-08 ***
## PT_time_medium:PT_fact_medium  0.118022   0.019074    6.188 7.36e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasibinomial family taken to be 0.02576204)
```

```
##
##     Null deviance: 92.197  on 2057  degrees of freedom
## Residual deviance: 52.523  on 2038  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 8
```

At least, all parameters left appear to be highly significant here. The model evaluation in a later section will show, how this model performs.

3 MODEL EVALUATION

This section assesses the model quality of the different models developed. The chapter is divided into a statistical part, which deals with key figures, and a graphical part, where the model quality can be assessed visually.

3.1 Model fitness

In this part, the statistical measuring of the RMSE and the Rsquare value is central. To do this, the following formula is formulated:

3.1.1 Function for Model Evaluation

For the evaluation of our models we will use the following function, which displays the RMSE and the R_squared for our predicted values.

Linear models and Generalized linear models with family binomial have slightly different properties in the measurement of model quality. For the linear models the adjusted R-squared value can be calculated additionally, for the GLM the explained variance can be calculated by the residual deviation and the null deviation.

```
eval_results <- function(true, predicted) {
  SSE <- sum((predicted - true)^2, na.rm = TRUE)
  SST <- sum((true - mean(true, na.rm = TRUE))^2, na.rm = TRUE)
  R_square <- 1 - SSE / SST
  RMSE = sqrt(SSE/length(true))

# Model performance metrics
  data.frame(RMSE = RMSE,
             Rsquare = R_square)
}

# Adjusted R square for linear model
adj_r2 <- function(lm_model) {
  summary(lm_model)$adj.r.squared
}

# deviance explained by glm:
dev_exp_glm <- function(glm_model) {
  with(summary(glm_model), 1 - deviance/null.deviance)
}
```

3.1.2 Model fitness of LM and GLM models:

The various measurements such as RMSE and Rsquare are now summarised in a table for all models. The table is defined as follows:

```
model_evaluation <- data.frame("Train_RMSE"=0, "Train_R^2"=0,
           "Test_RMSE"=0, "Test_R^2"=0, "nr_param"=0, "Dev.Expl_Adj.R2"=0)
```

The RMSE and Rsquare value is measured both for the Train and Test dataset individually, while the last field in the dataframe, "nr_param", describes the complexity of the model with the number of different parameters present.

The table is filled as the following, here with the example of the first GLM model for the GA data:

```
## Evaluating GLM 01 (original glm with all data)
model_evaluation["GA_GLM_01",] <- c(eval_results(d.norm.share.train$GA_share,
                                      predict(glm.share.01, d.norm.share.train,
                                              type="response")),
                                    eval_results(d.norm.share.test$GA_share,
                                      predict(glm.share.01, d.norm.share.test,
                                              type="response")),
                                    glm.share.01$rank,
                                    dev_exp_glm(glm.share.01))
```

The original data is compared with the prediction each for the test and Train dataset, using the above defined formula. With the "rank"-option, the number of parameters are filled in the last step. The same procedure is valid for all the other models including the ones for the HTA and FNT data. It is not showed here in order to save some space.

The filled table looks as the following:

```
round(model_evaluation, digits=2)
```

```
##                Train_RMSE Train_R.2 Test_RMSE Test_R.2 nr_param Dev.Expl_Adj.R2
## GA_GLM_01            0.01      0.82      0.01     0.80      593            0.80
## GA_GLM_03            0.02      0.26      0.02     0.29        4            0.27
## GA_GLM_04            0.02      0.51      0.02     0.52       33            0.51
## GA_GLM_05            0.02      0.49      0.02     0.51       23            0.50
## GA_LM_01_log         0.44      0.90      0.44     0.90       40            0.90
## GA_LM_01_cnt       234.89      0.88    268.03     0.97       40            0.90
## GA_LM_02_log         0.44      0.90      0.44     0.90       33            0.90
## GA_LM_02_cnt       234.08      0.88    179.46     0.99       33            0.90
## HTA_GLM_01           0.05      0.71      0.05     0.72       41            0.71
## HTA_GLM_02           0.06      0.59      0.06     0.56        8            0.59
## HTA_GLM_03           0.05      0.70      0.05     0.71       30            0.70
## HTA_GLM_04           0.05      0.69      0.05     0.71       24            0.69
## HTA_LM_01_log        0.21      0.98      0.23     0.97       40            0.97
## HTA_LM_01_cnt      548.67      0.97    868.23     0.99       40            0.97
## HTA_LM_02_log        0.21      0.98      0.23     0.97       30            0.97
## HTA_LM_02_cnt      564.35      0.97   1245.56     0.98       30            0.97
## FNT_GLM_01           0.03      0.49      0.03     0.46       41            0.45
## FNT_GLM_02           0.03      0.38      0.04     0.30        7            0.32
## FNT_GLM_03           0.03      0.48      0.03     0.41       32            0.43
## FNT_GLM_04           0.03      0.48      0.03     0.41       20            0.43
## FNT_LM_01_log        1.14      0.64      1.17     0.66       40            0.64
## FNT_LM_01_cnt     1404.51     -0.44   2354.18     0.70       40            0.64
## FNT_LM_02_log        1.14      0.64      1.17     0.66       32            0.64
## FNT_LM_02_cnt     1379.88     -0.39   2352.50     0.70       32            0.64
```

```
write.csv(model_evaluation, file="../Data/4_model_output/model_evaluation.csv")
```

Basically, it can be seen that the linear models have a higher accuracy, measured by the R-squared value. This is not surprising, however, since in the count data the population size plays a role in most variables

even after normalization, both in the explanatory and target variables. The more inhabitants a municipality has, the higher the number of tickets sold on average, as well as the number of people who are men, under 20 years of age, live in a one-person household or are divorced, for example. That is also the reason why in principle work was done with the shares. Nevertheless, the linear models were important to get an idea of the most important influencing factors through the step-by-step parameter elimination procedure with stepAIC.

Within the GLM for general subscriptions (GA), it is noticeable that the various models differ greatly in the number of parameters used. The model with all possible variables and interactions has almost 600 parameters, while the model after stepwise selection "GA_GLM_02" has only 4. The explained variance drops from 80% to 27% herewith. Even if only 4 variables can still explain a part, the loss of accuracy is too great. Hence the approach of reducing the original parameters with the stepwise selection from the linear model. In the end, 23 parameters were defined that had an R-squared value of exactly 50 %. This model "GA_GLM_05" builds the final model for the GA.

For the half-fare season tickets, the GLM model generally performs better with higher accuracies between 71% (all parameters after stepwise selection from the linear model) and 59% (model after stepwise selection within the GLM-binomial environment). Here one could say that the simplest model with only 8 parameters is accepted because it can still explain 59% of the variance. Since the procedure should be the same for all target variables, parameters were nevertheless added here again to reach the final model "HTA_GLM_04" with 24 parameters and an explained variance of 69%.

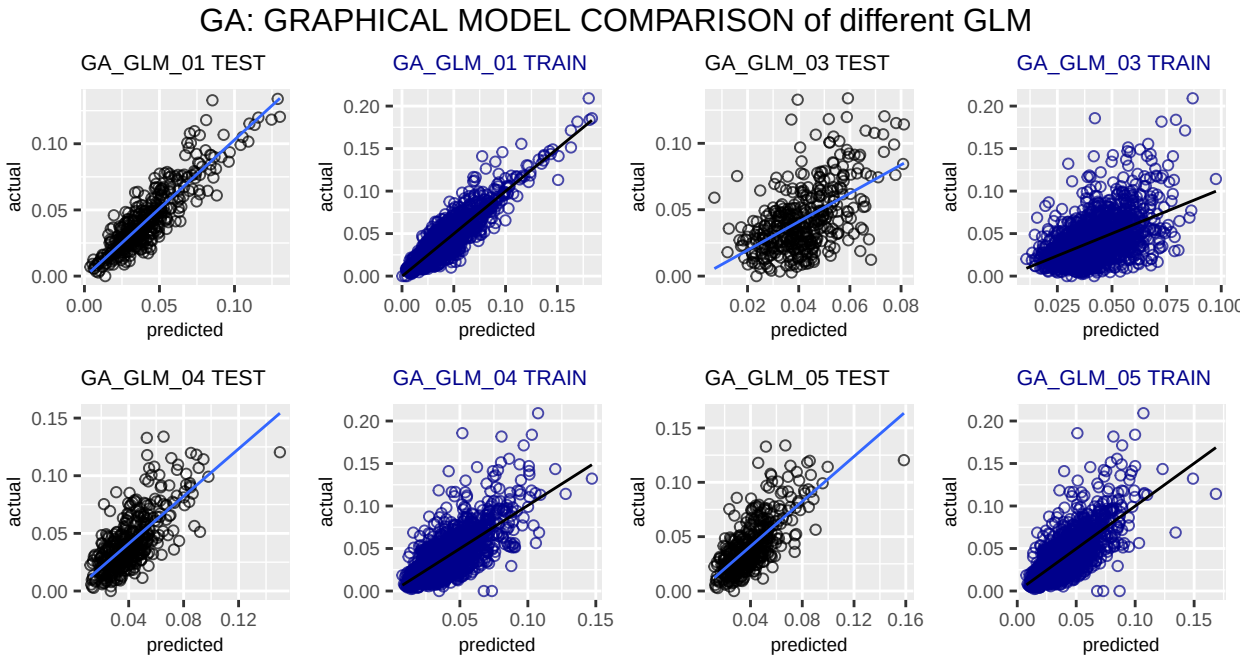
The procedure works least well for the Fare Network Tickets. With the same procedure, 43% of the variance can finally be explained with a model consisting of 20 parameters ("FNT_GLM_04"). Here it is also noticeable that the linear model with an R-squared value of 0.64 is clearly below the corresponding values for GA (0.90) and Half-Fare Travelcards (HTA, 0.97).

3.2 Graphical evaluation

The second part of the evaluation is a graphical analysis of the forecast values with the real values. This shows the characteristics of the different models and in particular graphically shows how strong the dispersion is compared to the true value.

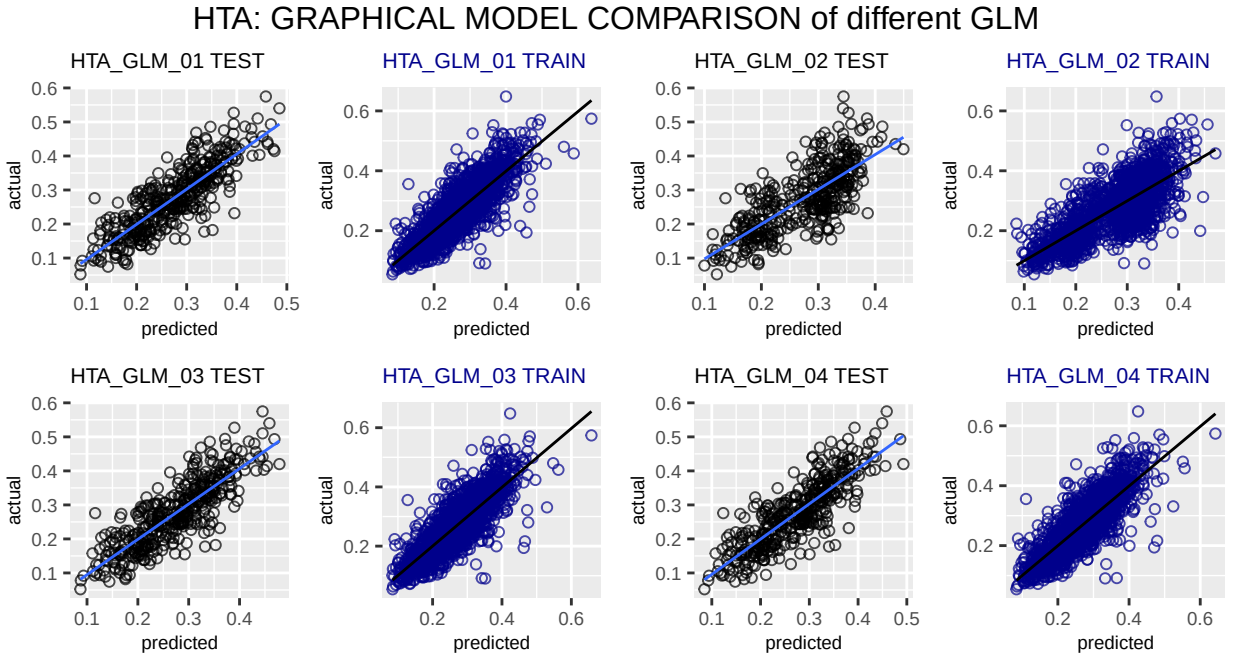
3.2.1 GA models

To start, the graphical evaluation of all different GLM models of the General Season Ticket (GA).



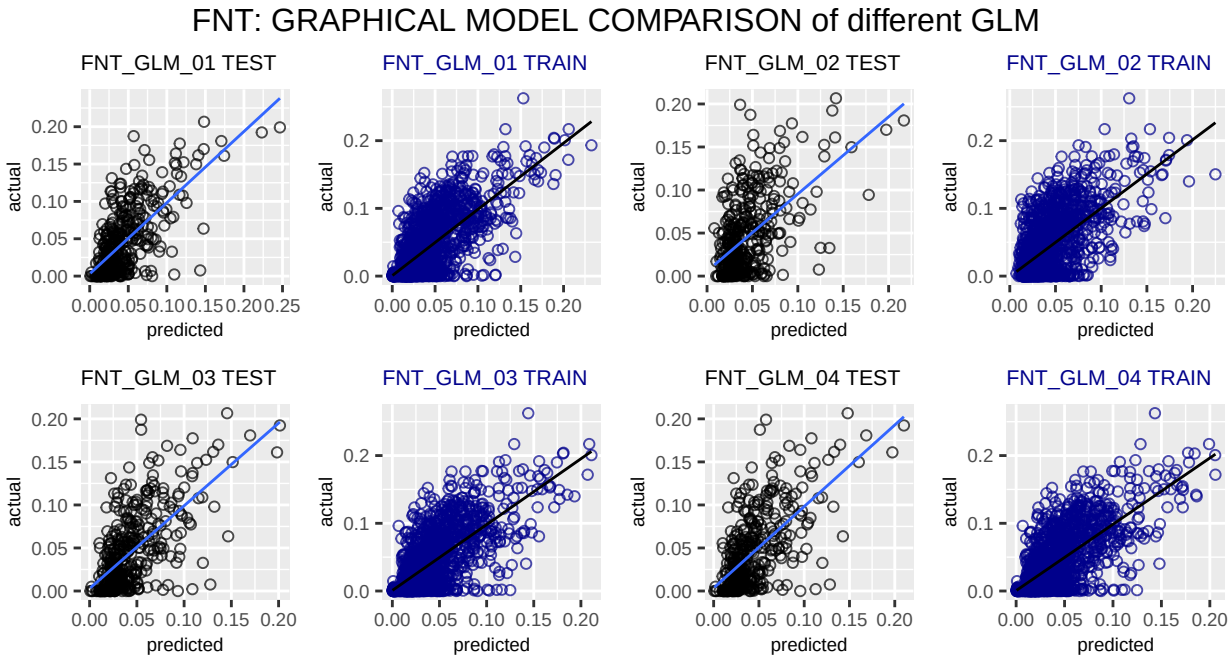
3.2.2 HTA models

And now all graphical evaluations of the different GLM models of the Half Fare Travelcards (HTA).



3.2.3 FNT models

Here the graphical evaluation of all different GLM models of the Fare Network Tickets (FNT).



3.3 Comparing estimates of the different models

Out of the graphical analysis, the final models can be confirmed as the following:

- GA_GLM_05
- HTA_GLM_04
- FNT_GLM_04

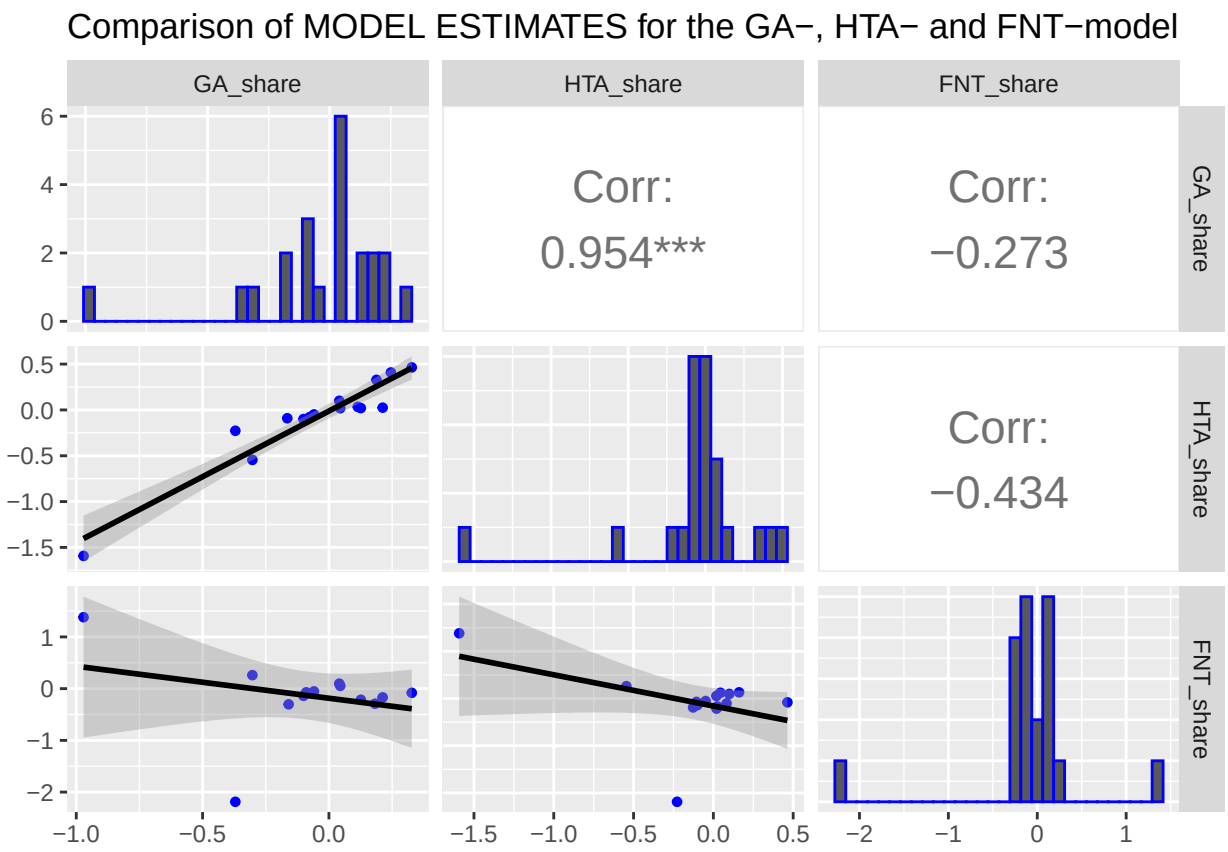
From these models, all estimates can be compared and stored in a separate table. As the all have normalized variables, the estimates can be compared as the weight is the same for all herewith.

```
ga.coef <- data.frame(as.list(glm.share.05$coefficients))

hta.coef <- data.frame(as.list(glm.HTA.share.04$coefficients))

fnt.coef <- data.frame(as.list(glm.FNT.share.04$coefficients))

inf_factors <- rbind.fill(ga.coef, hta.coef, fnt.coef)
inf_factors <- t(inf_factors)
colnames(inf_factors) <- c("GA_share", "HTA_share", "FNT_share")

### Write csv
write.csv(inf_factors, file="../Data/4_Model_output/inf_factors_estimates.csv",
          col.names=TRUE, row.names=TRUE)
```

```
## Warning in write.csv(inf_factors, file = "../Data/4_Model_output/
## inf_factors_estimates.csv", : attempt to set 'col.names' ignored
```

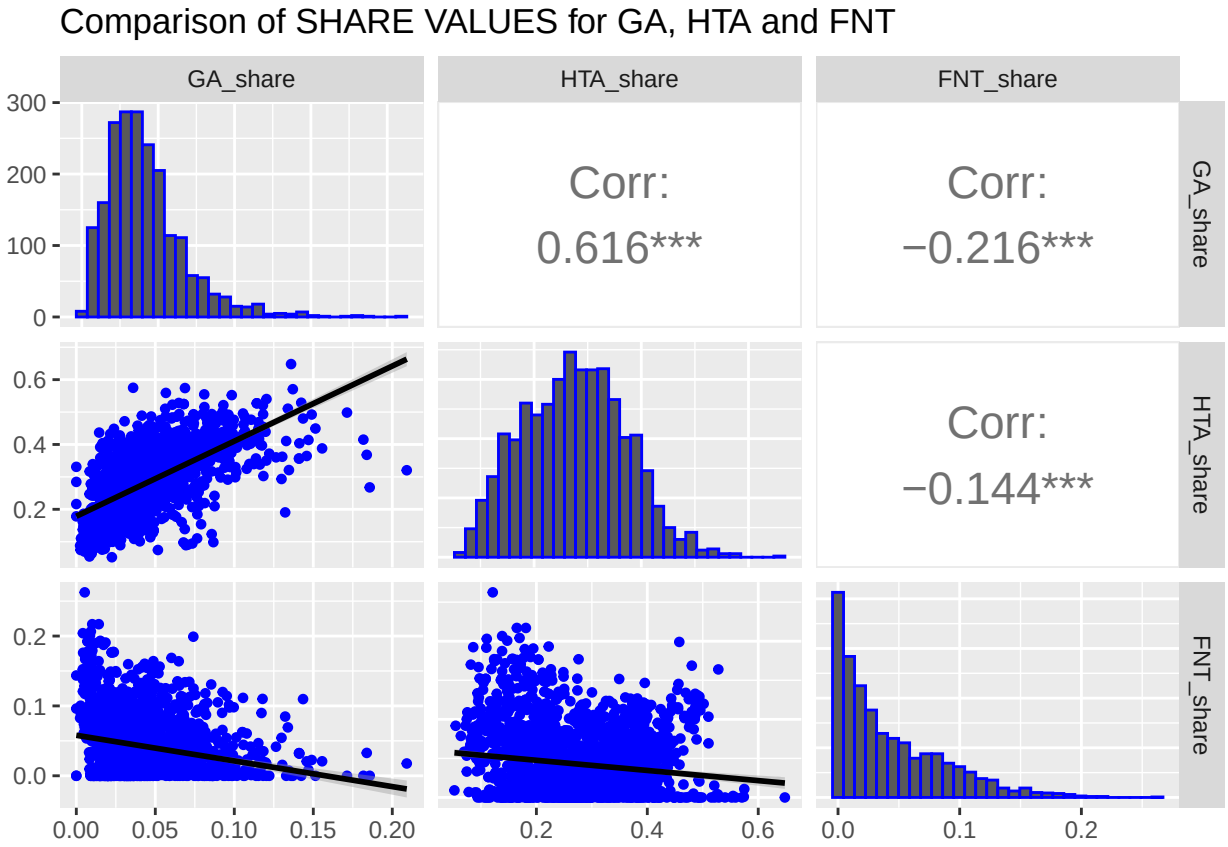
All estimates can be plotted against each other to recognize similarities between GA, HTA and FNT:

```
lowerFn <- function(data, mapping, method = "lm", ...) {
  p <- ggplot(data = data, mapping = mapping) +
    geom_point(colour = "blue", size=1.2) +
    geom_smooth(method = method, color = "black", ...)
  p
}


ggpairs(data.frame(inf_factors[2:32,]),
        lower = list(continuous= wrap(lowerFn, method = "lm")),
        diag = list(continuous = wrap("barDiag", colour = "blue")),
        upper = list(continuous = wrap("cor", size = 6))) +
  ggtitle("Comparison of MODEL ESTIMATES for the GA-, HTA- and FNT-model")
```



The estimates for GA and HTA are strongly correlating with a correlation value of 0.95! That means that the estimates for the both models are very similar, so high values in one estimate for the GA are correlating with high values in the same estimate for the HTA model. The FNT-model is behaving completely different with negative correlation values against the other two models. What, if we just look at the share values directly and compare them?

```
ggpairs(data.frame(d.share[,c("GA_share", "HTA_share", "FNT_share")]),
        lower = list(continuous= wrap(lowerFn, method = "lm")),
        diag = list(continuous = wrap("barDiag", colour = "blue")),
        upper = list(continuous = wrap("cor", size = 6))) +
  ggtitle("Comparison of SHARE VALUES for GA, HTA and FNT")
```



Similar correlation values can be found here as well. It seems that the proportions for GA and HTA behave very similarly. In municipalities with a high proportion of GA travelcards, half-fare cards are often also represented more frequently. On the other hand, it is the other way round with regard to the fare network tickets. The greater the proportion of GA or half-fare cards, the smaller the proportion of group season tickets tends to be.

3.4 Parameter selection for Cluster Analysis

The different estimates values of the individual parameters are shown in the Tableau Book "Influence factors".

All values in the table "inf_factors" are taken into the cluster analysis, which marks the next section of this script. For this purpose, a separate file is created, which reduces the original dataset to the relevant factors:

I build an extra dataframe with these variables, using the original share dataset:

```
rownames(inf_factors)
```

```
##  [1] "X.Intercept."          "age0_20"
##  [3] "birth_munic"           "birth_cant"
##  [5] "birth_CH"              "male"
##  [7] "resid_6_10y"           "hh_1"
##  [9] "hh_2"                  "hh_3_5"
## [11] "PT_time_medium"        "PT_time_big"
## [13] "train_stops_per_pop"   "bus_stat_per_1000"
## [15] "other_stat_per_1000"   "comb_car_1000"
## [17] "el_car_1000"           "languagefr"
```

```
## [19] "languageit"                    "languagerm"
## [21] "PT_fact_big"                   "PT_fact_big.PT_fact_medium"
## [23] "PT_time_medium.PT_time_big"    "single_share"
## [25] "married_share"                 "age20_40"
## [27] "other_stops_per_pop"           "train_stat_per_1000"
## [29] "PT_fact_medium"                "PT_fact_big.PT_time_big"
## [31] "bus_stops_per_pop"             "PT_time_medium.PT_fact_medium"
```

```
# filtering out language (loading separate) and interaction terms
cl.an.input <- d.norm.share[,c("language", rownames(inf_factors)
                               [c(2:17, 21, 24:29, 31)])]
```

4 CLUSTER ANALYSIS

The primary aim of cluster analysis is to identify spatial distribution patterns of certain characteristics with regard to relevant influencing factors. This requires the integration of the shares that were also used in the modelling. Particularly in cluster analysis, it is also important to carry out a normalization so that all variables have the same influence. Therefore, the normalized "share" table is used for the cluster analysis.

In this work, different approaches of cluster analysis are tried: On the one hand an agglomerative clustering model was used to start, which fits best an explorative approach to gain first insights (Ward, 1963). Further models were a k-means clustering (partitioning method) with a given number of clusters (MacQueen, 1967), a Generalized Mixture Model (GMM) which uses statistical models and is not based simply on a distance measure (Rasmussen, 1999), and the density-based clustering DBSCAN which assigns points to clusters based on densities in the data, returning also outliers (Ester et al., 1996).

4.1 Cluster analysis on the level of the municipality

In a first attempt, the cluster analysis is carried out at the level of the municipalities. Since this is an exploratory approach, primarily an agglomerative procedure is used (Ward, 1963). With the reasonable number of clusters obtained in this way, a k-means procedure is then applied later, along with density-based DBSCAN clustering as a supplement.

4.1.1 Preparation of clustering

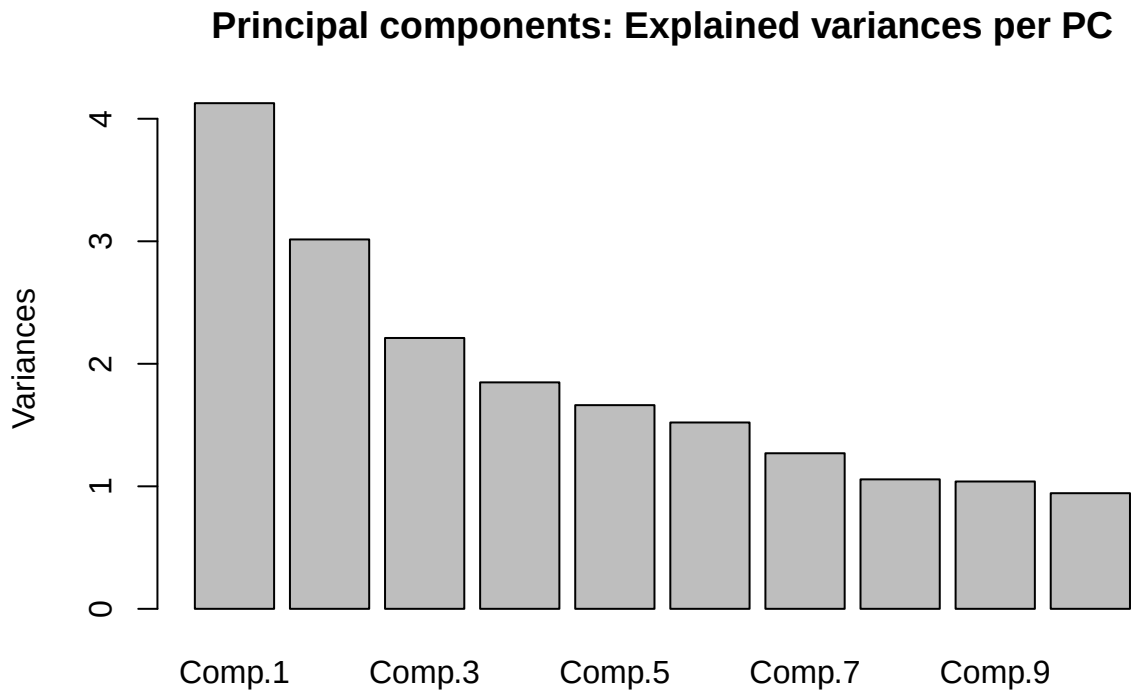
As a basis for all clustering methods, a distance matrix must be calculated. For a visual representation, the first two principal components will then finally also have to be calculated so that as much as possible of the variance present in the data can be represented two-dimensionally.

```
# Distance matrix
dp <- dist(cl.an.input[,2:25]) ## Euclidean distance

# Principal components
PC.inf.fac <- princomp(cl.an.input[,2:25]) # without language
summary(PC.inf.fac, loadings = FALSE)
```

```
## Importance of components:
##                            Comp.1    Comp.2     Comp.3    Comp.4     Comp.5
## Standard deviation     2.0315533 1.7363261 1.48685577 1.3595490 1.28935785
## Proportion of Variance 0.1720506 0.1256789 0.09215895 0.0770530 0.06930216
## Cumulative Proportion  0.1720506 0.2977295 0.38988850 0.4669415 0.53624366
##                            Comp.6    Comp.7     Comp.8    Comp.9    Comp.10
## Standard deviation     1.23324400 1.1268066 1.02795786 1.0194851 0.97141932
## Proportion of Variance 0.06340126 0.0529296 0.04405046 0.0433273 0.03933809
## Cumulative Proportion  0.59964492 0.6525745 0.69662498 0.7399523 0.77929037
##                           Comp.11    Comp.12    Comp.13    Comp.14    Comp.15
## Standard deviation     0.8596227 0.85045749 0.83176850 0.78354432 0.71261488
## Proportion of Variance 0.0308046 0.03015123 0.02884063 0.02559334 0.02116945
## Cumulative Proportion  0.8100950 0.84024620 0.86908684 0.89468018 0.91584963
##                           Comp.16    Comp.17    Comp.18   Comp.19     Comp.20
## Standard deviation     0.65673564 0.63843179 0.60576430 0.5085122 0.455547631
## Proportion of Variance 0.01797964 0.01699139 0.01529703 0.0107796 0.008651022
## Cumulative Proportion  0.93382927 0.95082066 0.96611769 0.9768973 0.985548309
##                            Comp.21     Comp.22     Comp.23      Comp.24
```

```
## Standard deviation     0.413050499 0.294130248 0.283472945 0.0958739795
## Proportion of Variance 0.007112236 0.003606444 0.003349832 0.0003831787
## Cumulative Proportion  0.992660545 0.996266989 0.999616821 1.0000000000
```

```r
# first 2 PCs explain approx. 30% of the variance, 5 PC about 54%

plot(PC.inf.fac, main = "Principal components: Explained variances per PC")
```

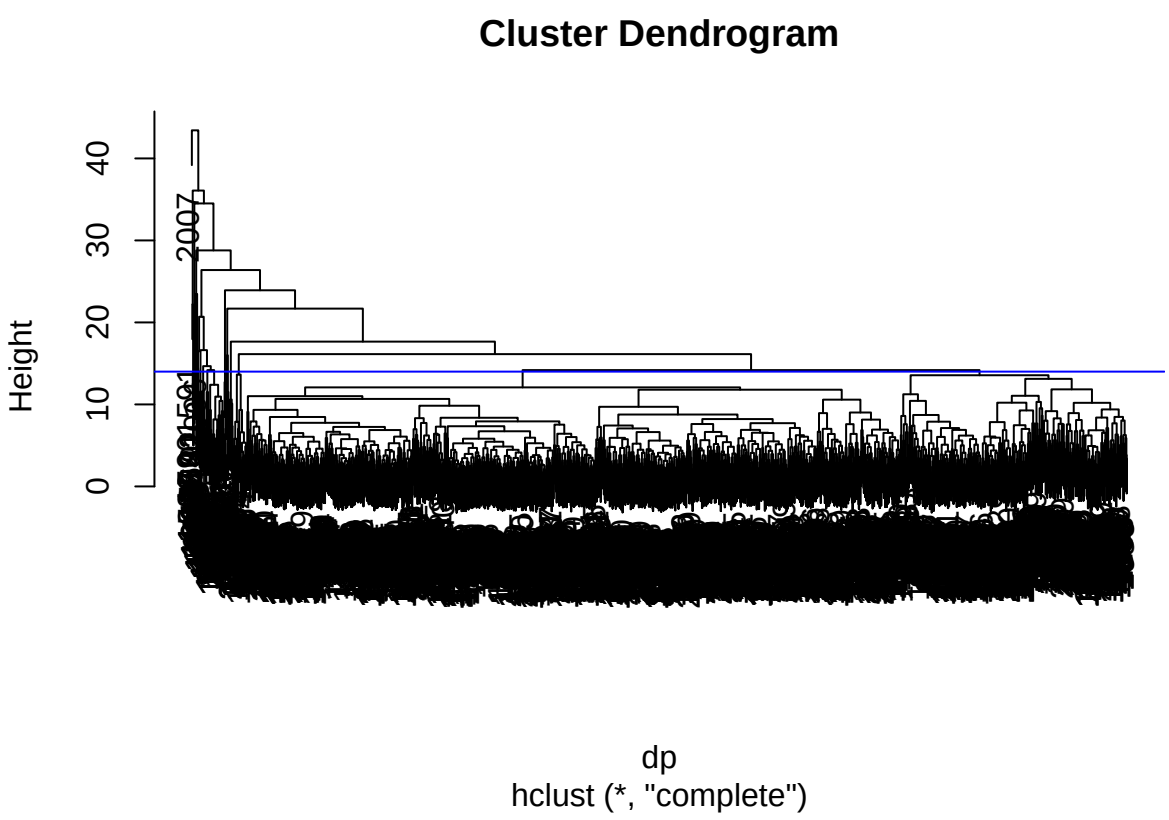



```r
# write two first principal components out
pc <- PC.inf.fac$scores[,1:2]
```

4.1.2 Agglomerative Clustering

First, as described above, the approach of agglomerative clustering. This fits best the explorative purpose of this work.

```r
## Agglomerative clustering with complete linkage
cl.inf.fac <- hclust(dp, method = "complete")

## Show dendrogram
plot(cl.inf.fac)
abline(h=14, col="blue")
```

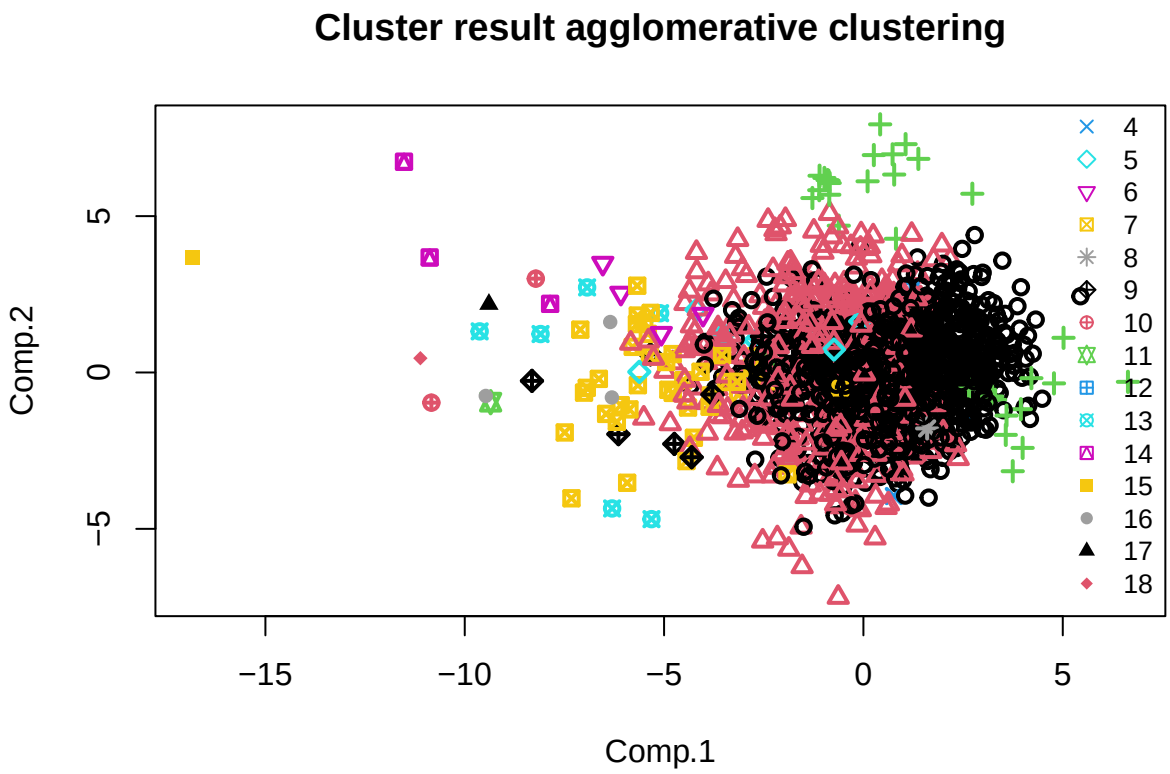


Out of the dendrogram, it is hard to say how many clusters would be accurate. But it can be seen, that it's going to be very hard to recognize clusters, as there are many clusters with only very little observations which are cut very far up in the tree. I try to cut the tree approximately on the drawn line so that I get at least 2 larger groups.

```
## Split into groups
k = 18 # with 18 clusters, at least 2 big groups!
grps <- cutree(cl.inf.fac, k = k)

table(grps)
```

```
## grps
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 1437  499   30    2    9    4   41    5    8    2    2    4    6    3    1    3
##   17   18
##    1    1
```

```
## Visualize result in PC 1 & 2
plot(pc, pch = grps, col=grps, lwd=2, main="Cluster result agglomerative clustering")
legend("bottomright", legend = 1:k, pch = 1:k, col=1:k, bty="n", cex=0.85)
```




```
## Look at means of clusters / centroids
# aggregate(cl.an.input, by=list(cluster=grps), mean)
```

The silhouette values can give us an evluation of the quality:

```
## Silhouette plot from package "cluster"
summary(cluster::silhouette(grps, dp))
```

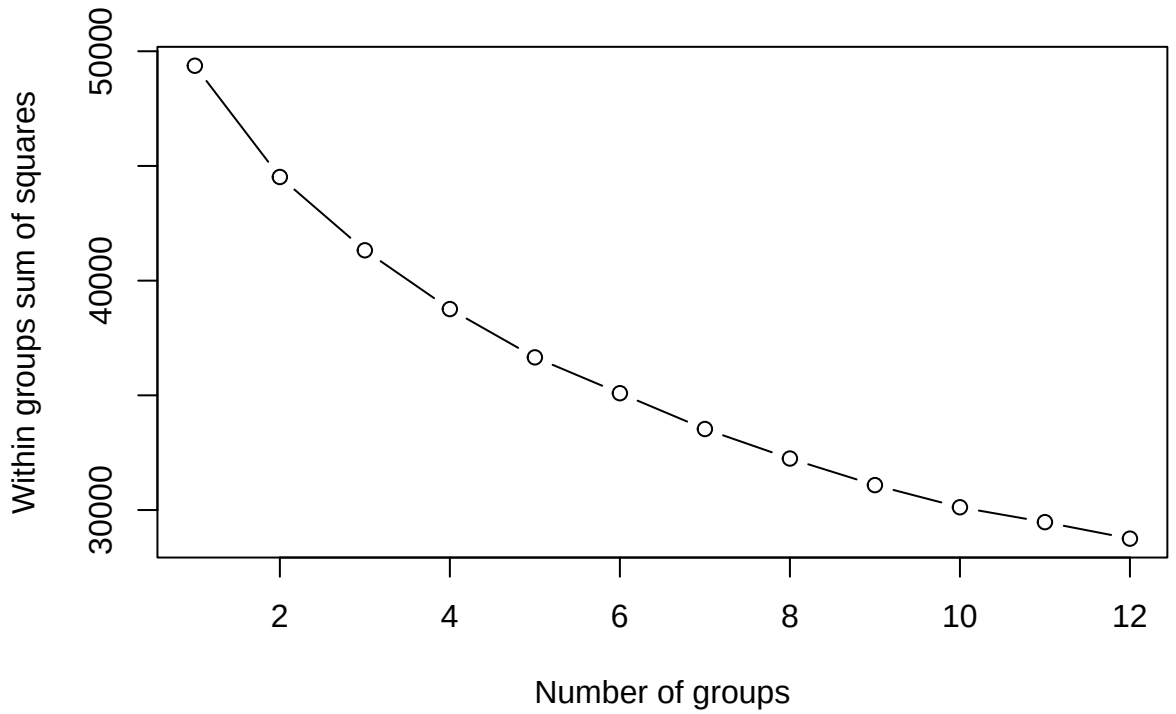
```
## Silhouette of 2058 units in 18 clusters from silhouette.default(x = grps, dist = dp) :
##  Cluster sizes and average silhouette widths:
##        1437         499          30           2           9           4
##  0.17211056 -0.05953620  0.03479200  0.16262482  0.25126546  0.14805391
##          41           5           8           2           2           4
##  0.03999173  0.21953811  0.16017068  0.39476326 -0.08194178  0.40659218
##           6           3           1           3           1           1
##  0.13357709  0.16971259  0.00000000  0.23201081  0.00000000  0.00000000
## Individual silhouette widths:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.2762  0.0212  0.1486  0.1118  0.2064  0.5811
```

Due to the amount of data, the silhouette plot is not shown properly, but the average silhouette width of 0.11 is very low, making the classification relatively bad.

4.1.3 k-means clustering

The second method is k-means clustering, in which it must be defined from the beginning how many clusters there should eventually be.

```
# k-means

## Choose number of clusters k with scree plot
nr <- 12
wss <- rep(0, nr)
for (i in 1:nr) wss[i] <- sum(kmeans(cl.an.input[,2:25], centers = i, nstart = 20)$withinss)
par(mfrow = c(1,1))
plot(1:nr, wss, type = "b", xlab = "Number of groups", ylab = "Within groups sum of squares")
```



No kink can be detected here unfortunately. With this scree plot, there is no good basis for a decision here, so everything from 1-10 is tried out and the corresponding silhouette values are compared with each other.

```
seed = 123
for(k in 2:10) {
  seed = 123
  ## k-means with 3-10 centers
  ckm <- kmeans(cl.an.input[,2:25], centers = k, nstart = 10)
  grpsKM <- ckm$cluster

  ## Silhouette Summary
  print(paste("k means clustering with ", k, "clusters: Individual silhouette widths:"))
```

```
  print(summary(silhouette(grpsKM, dp))[1])

}
```

```
## [1] "k means clustering with  2 clusters: Individual silhouette widths:"
## $si.summary
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.2325  0.1842  0.2920  0.2258  0.3429  0.4392
##
## [1] "k means clustering with  3 clusters: Individual silhouette widths:"
## $si.summary
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.26878  0.02135  0.08763  0.07922  0.15548  0.30615
##
## [1] "k means clustering with  4 clusters: Individual silhouette widths:"
## $si.summary
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.31588  0.03105  0.08946  0.08690  0.15529  0.30917
##
## [1] "k means clustering with  5 clusters: Individual silhouette widths:"
## $si.summary
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.22250  0.03067  0.09175  0.09004  0.16095  0.32863
##
## [1] "k means clustering with  6 clusters: Individual silhouette widths:"
## $si.summary
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.41067  0.03051  0.09175  0.08766  0.15606  0.29885
##
## [1] "k means clustering with  7 clusters: Individual silhouette widths:"
## $si.summary
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.25667  0.02002  0.10010  0.08802  0.16219  0.41116
##
## [1] "k means clustering with  8 clusters: Individual silhouette widths:"
## $si.summary
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.26396  0.03166  0.08642  0.08897  0.15309  0.41460
##
## [1] "k means clustering with  9 clusters: Individual silhouette widths:"
## $si.summary
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.25885  0.02140  0.07337  0.07410  0.13139  0.41240
##
## [1] "k means clustering with  10 clusters: Individual silhouette widths:"
## $si.summary
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.22411  0.03370  0.09173  0.09246  0.15916  0.40185
```

Possibilities are k=2 or k=6 here:

```r
## k-means with 2 centers
ckm2 <- kmeans(cl.an.input[,2:25], centers = 2, nstart = 10)
grpsKM2<- ckm2$cluster

## k-means with 6 centers
ckm6 <- kmeans(cl.an.input[,2:25], centers = 6, nstart = 10)
grpsKM6<- ckm6$cluster
```

4.1.4 Gaussian mixture model

The Guassian mixture model makes the assumption that there is a statistical model behind the data. In this case, that is also partly the case, which makes it worthwhile to try it out.

```r
################################
## Gaussian Mixture Models ####
################################
mc <- Mclust(cl.an.input[,2:25]) # takes long time!
table(mc$classification)
```

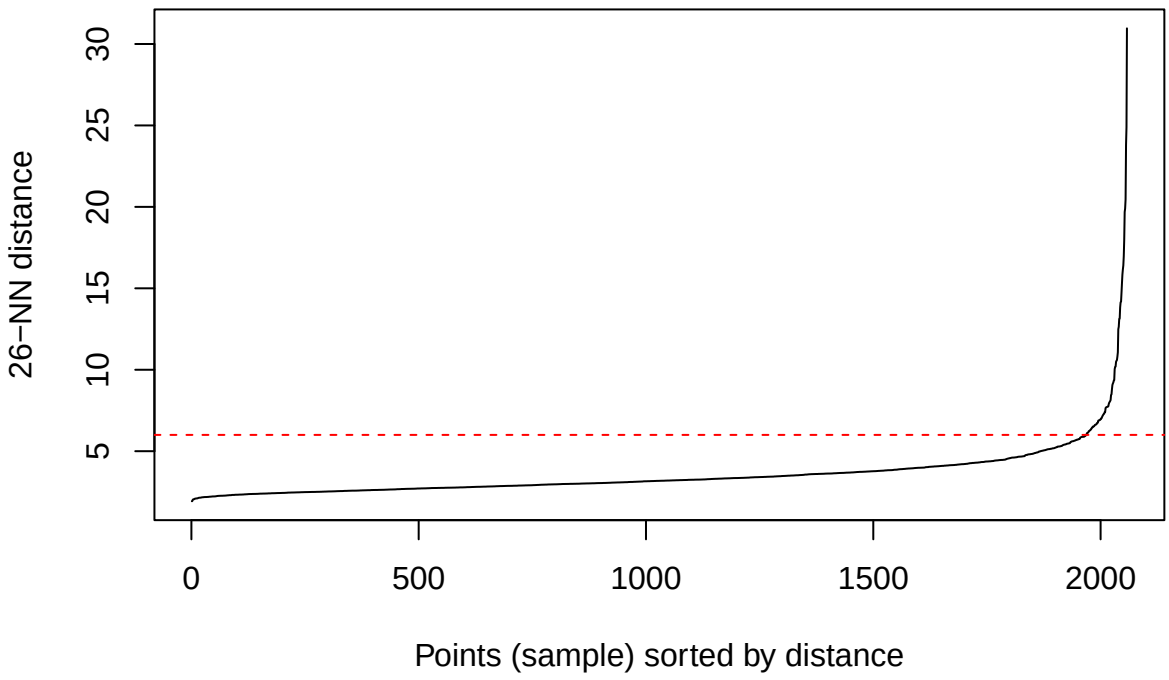
```
##
##   1   2   3   4   5   6   7
## 307 166 611 459 293 143  79
```

With this approach, 7 classes were built here.

4.1.5 DBSCAN

The last approach is the "Density-Based Spatial Clustering of Applications with Noise". Two characteristics distinguish this procedure: On the one hand, it is density-based, i.e. it sees points that are close to each other from a density-based perspective as belonging together. On the other hand, it classifies points that lie outside the recognised clusters as "noise" points. This means that the method can deal well with outliers by simply deleting them from the cluster analysis. In the case of the many municipalities, there are certainly outliers, which also makes the procedure make sense here.

```r
# DBSCAN
kNNdistplot(cl.an.input[,2:25], k = 26) ## -> k = dim + 1; eps value = 6 (sharp increase)
abline(h=6, col = "red", lty=2)
```




```
# setting parameters according to minPts = k and eps = sharp increase
minPts=26 # = k
eps=6 # cutting line above
dbs <- dbscan(cl.an.input[,2:25], eps = eps, minPts = minPts)
max(dbs$cluster) # 1
```

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

Only 1 cluster is present here, alternate values of minPts and eps to get another amount of clusters:

```
# Searching for better parameters:
## For eps value (i), only values from 1 to 5 are tested, because higher values
## always lead to only one cluster at the end.

# For minPts, the range from 2 to 5 is tested. 1 would lead to as many clusters
# as points are and for more than 5, only one cluster can be obtained.

for(i in (1:5)) {

  for(j in (2:5)) {

    dbs <- dbscan(cl.an.input[,2:25], eps = i, minPts = j)

    print(paste("minPts:", j, "; eps: ", i, "; nr. of clusters: ", max(dbs$cluster))) # 1
    print(table(dbs$cluster))
```

```
  }

}
```

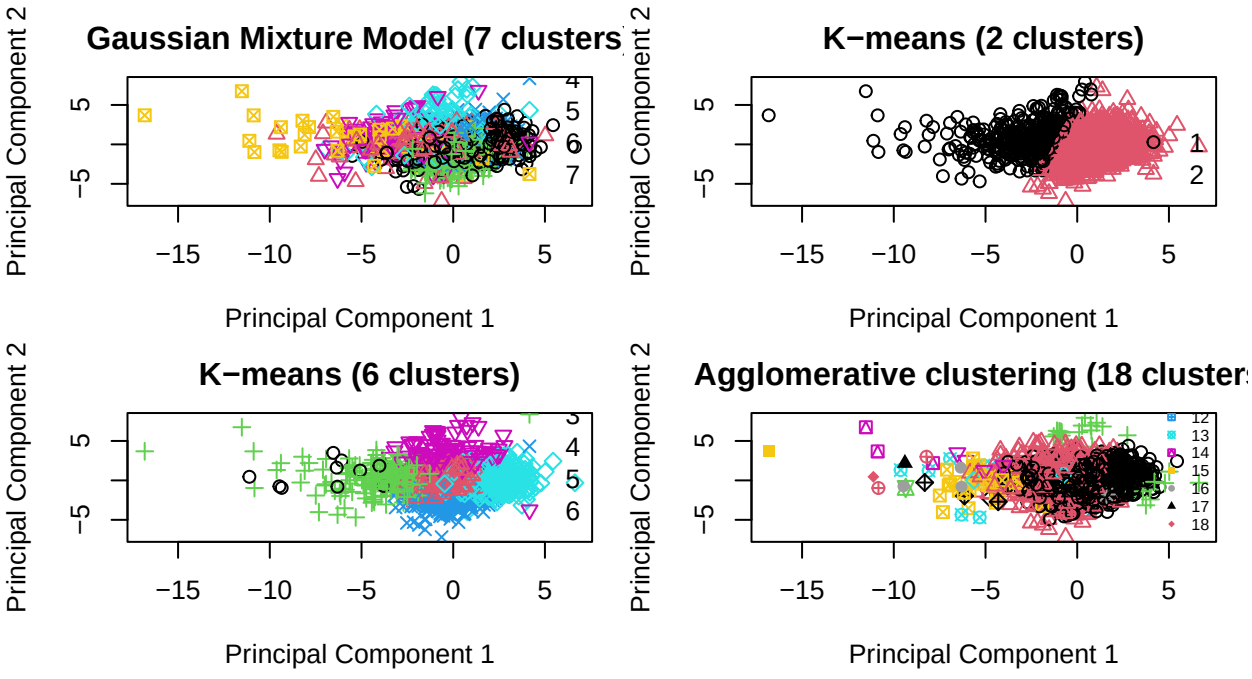
```
## [1] "minPts: 2 ; eps:  1 ; nr. of clusters:  0"
##
##    0
## 2058
## [1] "minPts: 3 ; eps:  1 ; nr. of clusters:  0"
##
##    0
## 2058
## [1] "minPts: 4 ; eps:  1 ; nr. of clusters:  0"
##
##    0
## 2058
## [1] "minPts: 5 ; eps:  1 ; nr. of clusters:  0"
##
##    0
## 2058
## [1] "minPts: 2 ; eps:  2 ; nr. of clusters:  83"
##
##    0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
## 1231    4  585    3    5    2    2    2    2    2    2    2    3    2    2    2
##   16   17   18   19   20   21   22   23   24   25   26   27   28   29   30   31
##    9    3    2    6    2    2    3    2    4    5    2    2    3    3    5    2
##   32   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47
##   11    5    2    3    4    2    2    3    2    2    2    2    3    2    3    2
##   48   49   50   51   52   53   54   55   56   57   58   59   60   61   62   63
##    2    2    3    2    2    4    2    2    2    2    3    2    4   14    2    4
##   64   65   66   67   68   69   70   71   72   73   74   75   76   77   78   79
##    4    3    2    2    2    2    2    3    6    2    3    2    2    2    3    2
##   80   81   82   83
##    2    2    2    2
## [1] "minPts: 3 ; eps:  2 ; nr. of clusters:  33"
##
##    0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
## 1331    4  585    5    3    4    5    9    3    3    3   11    5    6    3    3
##   16   17   18   19   20   21   22   23   24   25   26   27   28   29   30   31
##    3    3    3    3    4    4    3    3   14    4    4    3    3    6    4    3
##   32   33
##    3    5
## [1] "minPts: 4 ; eps:  2 ; nr. of clusters:  17"
##
##    0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
## 1443  534    5    4    8    3   11    6    4    4    4    4    6    3    4    6
##   16   17
##    4    5
## [1] "minPts: 5 ; eps:  2 ; nr. of clusters:  5"
##
##    0    1    2    3    4    5
## 1533  496    5    7   11    6
## [1] "minPts: 2 ; eps:  3 ; nr. of clusters:  16"
```

```
##
##    0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
##  353 1667    2    3    2    2    2    2    4    2    4    3    3    2    3    2
##   16
##    2
## [1] "minPts: 3 ; eps:  3 ; nr. of clusters:  7"
##
##    0    1    2    3    4    5    6    7
##  371 1667    3    4    3    3    4    3
## [1] "minPts: 4 ; eps:  3 ; nr. of clusters:  1"
##
##    0    1
##  397 1661
## [1] "minPts: 5 ; eps:  3 ; nr. of clusters:  1"
##
##    0    1
##  411 1647
## [1] "minPts: 2 ; eps:  4 ; nr. of clusters:  3"
##
##    0    1    2    3
##  132 1922    2    2
## [1] "minPts: 3 ; eps:  4 ; nr. of clusters:  1"
##
##    0    1
##  136 1922
## [1] "minPts: 4 ; eps:  4 ; nr. of clusters:  1"
##
##    0    1
##  140 1918
## [1] "minPts: 5 ; eps:  4 ; nr. of clusters:  1"
##
##    0    1
##  142 1916
## [1] "minPts: 2 ; eps:  5 ; nr. of clusters:  2"
##
##    0    1    2
##   59 1997    2
## [1] "minPts: 3 ; eps:  5 ; nr. of clusters:  1"
##
##    0    1
##   61 1997
## [1] "minPts: 4 ; eps:  5 ; nr. of clusters:  1"
##
##    0    1
##   62 1996
## [1] "minPts: 5 ; eps:  5 ; nr. of clusters:  1"
##
##    0    1
##   64 1994
```

DBSCAN works very poorly here. Either practically all points are noise points or in a cluster, there is nothing in between.

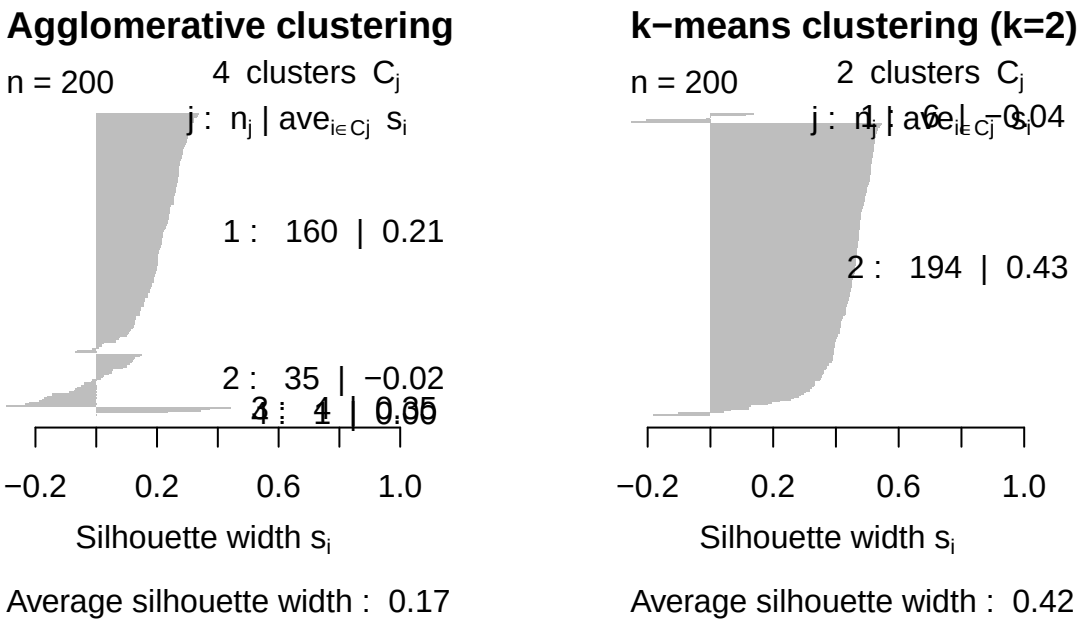
4.1.6 Comparison of different approaches

```
## PLOT CLUSTER CALSSES ON PC 1 & 2

par(mfrow=c(2,2), mar=c(4,4,2.5,1))

# Gaussian Mixture model
plot(pc, pch = mc$classification, col = mc$classification,
     main="Gaussian Mixture Model (7 clusters)",
       xlab="Principal Component 1", ylab="Principal Component 2")
legend("bottomright", legend = 1:7, pch = 1:7, col=1:7, bty="n")

# K-means with 2 clusters
plot(pc, pch = grpsKM2, col=grpsKM2, main="K-means (2 clusters)",
       xlab="Principal Component 1", ylab="Principal Component 2")
legend("bottomright", legend = 1:2, pch = 1:2, col=1:2, bty="n")

# K-means with 6 clusters
plot(pc, pch = grpsKM6, col=grpsKM6, main="K-means (6 clusters)",
       xlab="Principal Component 1", ylab="Principal Component 2")
legend("bottomright", legend = 1:6, pch = 1:6, col=1:6, bty="n")

# Agglomerative clustering
plot(pc, pch = grps, col=grps, main="Agglomerative clustering (18 clusters)",
       xlab="Principal Component 1", ylab="Principal Component 2")
legend("bottomright", legend = 1:18, pch = 1:18, col=1:18, bty="n", cex=0.56)
```

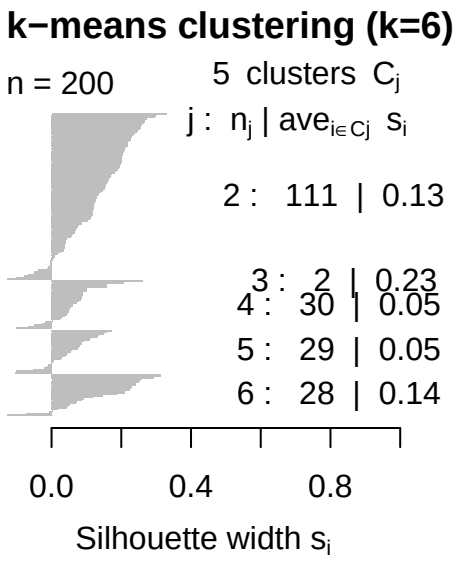


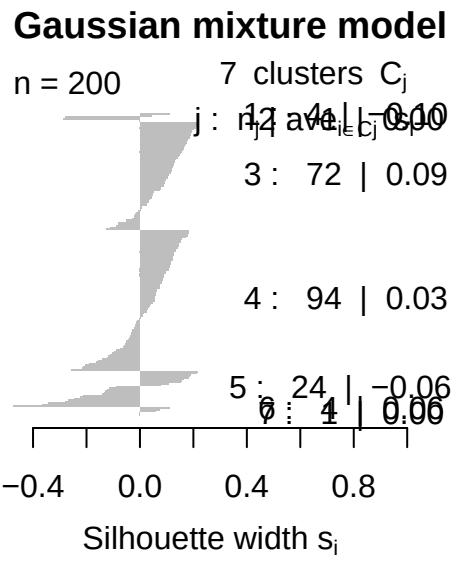

```
## SILHOUETTE PLOTS ##

## Reduce number of data to see a sample of the whole dataset
```

```
# classifications
grps.ss <- grps[1:200]
mc.ss <- mc$classification[1:200]
grpsKM2.ss <- grpsKM2[1:200]
grpsKM6.ss <- grpsKM6[1:200]

# distance table
dp.ss <- dist(cl.an.input[1:200,2:25]) ## Euclidean

# Silhouette Plot from package "cluster"
par(mfrow=c(1,2), mar=c(5,4,4,2))


plot(silhouette(grps.ss, dp.ss, cex=0.8), main="Agglomerative clustering")
plot(silhouette(grpsKM2.ss, dp.ss), main="k-means clustering (k=2)")
```




```
plot(silhouette(grpsKM6.ss, dp.ss), main="k-means clustering (k=6)")
plot(silhouette(mc.ss, dp.ss), main="Gaussian mixture model")
```






```
table(grps)
```

```
## grps
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 1437  499   30    2    9    4   41    5    8    2    2    4    6    3    1    3
##   17   18
##    1    1
```

```
table(mc$classification)
```

```
##
##   1   2   3   4   5   6   7
## 307 166 611 459 293 143  79
```

```
table(grpsKM2)
```

```
## grpsKM2
##    1    2
##  457 1601
```

```
table(grpsKM6)
```

```
## grpsKM6
##   1   2   3   4   5   6
##  10 714 211 435 443 245
```

```
summary(silhouette(grps, dp))[1]
```

```
## $si.summary
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.2762  0.0212  0.1486  0.1118  0.2064  0.5811
```

```
summary(silhouette(grpsKM2, dp))[1]
```

```
## $si.summary
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.2325  0.1842  0.2920  0.2258  0.3429  0.4392
```

```
summary(silhouette(grpsKM6, dp))[1]
```

```
## $si.summary
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.29569  0.02617  0.09428  0.08483  0.15706  0.39877
```

```
summary(silhouette(mc$classification, dp))[1]
```

```
## $si.summary
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.613780 -0.098573  0.007345 -0.012914  0.089425  0.250841
```

```
## GROUPS REPRESENTING LANGUAGE REGIONS

ca.aggl.lan <- data.frame(cls = grps, lang = cl.an.input[,1])
table(ca.aggl.lan)
```

```
##     lang
## cls   de  fr  it  rm
##   1  895 463  66  13
##   2  337 109  31  22
##   3   11  19   0   0
##   4    2   0   0   0
##   5    5   4   0   0
##   6    4   0   0   0
##   7   17   2  15   7
##   8    3   2   0   0
##   9    6   0   2   0
##   10   1   1   0   0
##   11   1   0   1   0
##   12   3   0   0   1
##   13   3   0   2   1
##   14   2   0   1   0
##   15   0   0   1   0
##   16   3   0   0   0
##   17   1   0   0   0
##   18   1   0   0   0
```

```
ca.km2.lan <- data.frame(cls = grpsKM2, lang = cl.an.input[,1])
table(ca.km2.lan)
```

```
##    lang
## cls   de   fr   it   rm
##   1  241   85   95   36
##   2 1054  515   24    8
```

```
ca.km6.lan <- data.frame(cls = grpsKM6, lang = cl.an.input[,1])
table(ca.km6.lan)
```

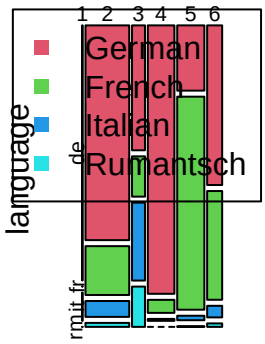
```
##    lang
## cls  de  fr  it  rm
##   1   9   0   1   0
##   2 541 123  40  10
##   3  93  30  58  30
##   4 412  20   3   0
##   5 102 333   7   1
##   6 138  94  10   3
```

```
ca.mc.lan <- data.frame(cls = mc$classification, lang = cl.an.input[,1])
table(ca.mc.lan)
```

```
##    lang
## cls  de  fr  it  rm
##   1  64 237   0   6
##   2  73  66  16  11
##   3 504 107   0   0
##   4 366  92   0   1
##   5 149  58  80   6
##   6  78  31  18  16
##   7  61   9   5   4
```

```
par(mar=c(5.1, 4.1, 4.1, 8.1), xpd=TRUE)
plot(table(ca.km6.lan), col=c(2:5), main="language distribution on clusters", xlab="cluster nr.", ylab=
legend("topright", inset=c(-0.2, 0), legend = c("German", "French", "Italian", "Rumantsch"), col=c(2:5)
```

guage distribution on clusters



The outcome of the cluster analysis on the municipality level is, that it is not applicable, as the huge amount of data is too wide-spreaded. The best approach is reached with a k-means clustering with k=6. This can be written in a separate table to look at it graphically.

4.1.7 Exporting enriched data

The generated clusters can now be written into a table together with the relevant influencing factors, which can then be used for the distribution of shares and graphical analysis with Tableau.

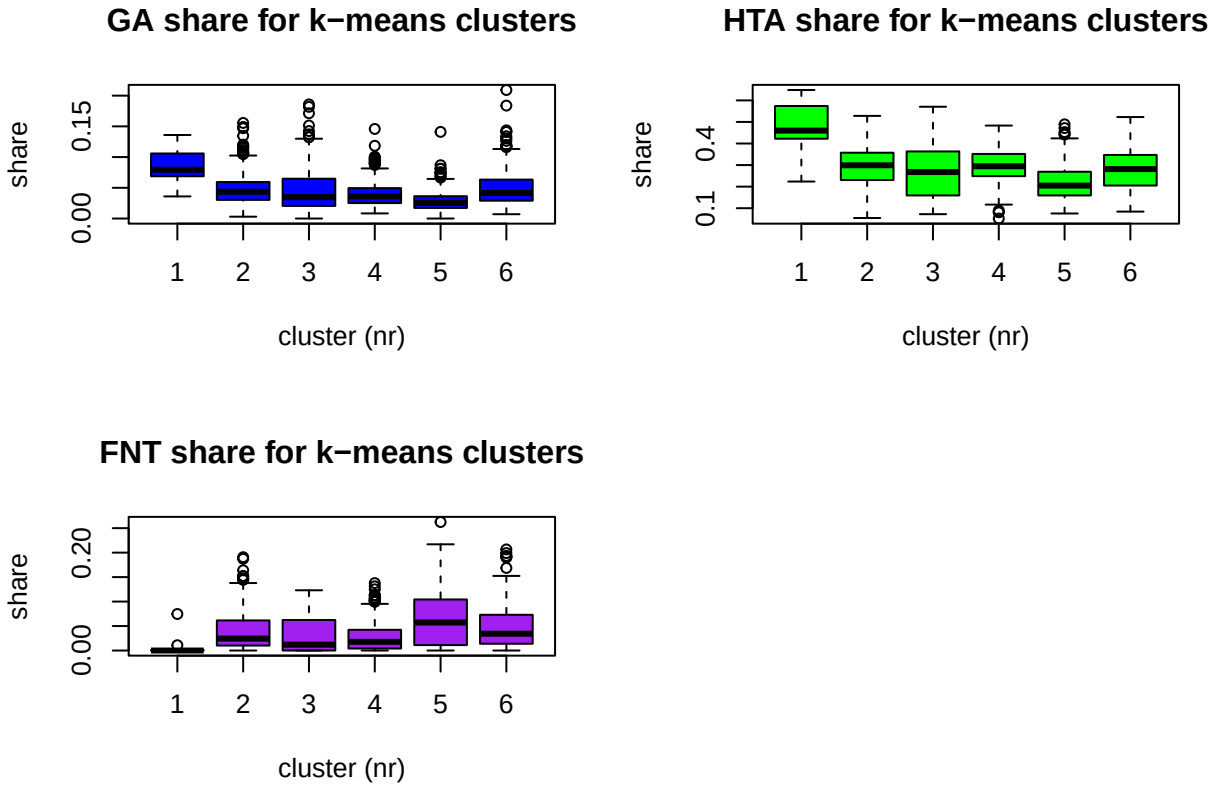
```
# filtering out language (loading separate) and interaction terms
d.share.clust <- d.share[,c("language", rownames(inf_factors)[c(2:17, 21, 24:29, 31)])]

d.share.clust["k_cluster"] = grpsKM6

d.share.clust["GA_share"] = d.share$GA_share
d.share.clust["HTA_share"] = d.share$HTA_share
d.share.clust["FNT_share"] = d.share$FNT_share

d.share.clust["BFS_Nr"] = d.share$BFS_Nr
d.share.clust["municipality"] = d.share$municipality
d.share.clust["canton"] = d.share$canton
d.share.clust["language"] = d.share$language
```

```
write.csv(d.share.clust, file="../Data/4_Model_output/influence_factors_with_CA.csv")
```

4.1.8 Share distribution of GA, HTA and FNT for clusters

```
par(mfrow=c(2,2))
boxplot(d.share.clust$GA_share ~ d.share.clust$k_cluster, col="blue",
```

```
        main = "GA share for k-means clusters", xlab="cluster (nr)", ylab="share")
boxplot(d.share.clust$HTA_share ~ d.share.clust$k_cluster, col="green",
        main = "HTA share for k-means clusters", xlab="cluster (nr)", ylab="share")
boxplot(d.share.clust$FNT_share ~ d.share.clust$k_cluster, col="purple",
        main = "FNT share for k-means clusters", xlab="cluster (nr)", ylab="share")
```



4.2 Cluster analysis on the level of the cantons

4.2.1 Loading and normalizing data

```
# Loading data
d.cant.share <- read.csv("../Data/3_Output/inf_fac_cant_share.csv")
# print(colnames(d.cant.share))

# select columns to normalize
columns <- c("PT_dist_medium", "PT_time_medium", "PT_dist_big", "PT_time_big",
             "str_dist_medium", "str_time_medium" ,"str_dist_big", "str_time_big",
             "PT_fact_big", "PT_fact_medium", "single_share", "married_share",
             "widowed_share", "divorced_share", "GA_share", "HTA_share",
             "FNT_share", "age0_20_share", "age20_40_share", "age40_60_share",
             "age60._share", "birth_munic_share", "birth_cant_share",
             "birth_CH_share", "birth_notCH_share", "male_share", "female_share",
             "resid_0_1y_share", "resid_1_5y_share", "resid_6_10y_share",
```

```
             "resid_10.y_share", "hh_1_share", "hh_2_share", "hh_3_5_share",
             "hh_6._share", "bus_stops_per_pop", "other_stops_per_pop",
             "train_stops_per_pop", "bus_stat_per_1000", "other_stat_per_1000",
             "train_stat_per_1000", "comb_car_per_1000", "el_car_per_1000")

# normalize data
d.cant.norm.share <- d.cant.share
d.cant.norm.share[columns] <- scale(d.cant.norm.share[columns])
```

```
cl.an.cant.input <- d.cant.norm.share[,c("age0_20_share", "age20_40_share",
                      "birth_munic_share", "birth_cant_share", "birth_CH_share",
                      "male_share", "resid_6_10y_share", "hh_1_share", "hh_2_share",
                      "hh_3_5_share", "PT_time_medium", "PT_time_big", "train_stops_per_pop",
                      "bus_stat_per_1000", "other_stat_per_1000", "comb_car_per_1000",
                      "el_car_per_1000", "PT_fact_big", "single_share", "married_share",
                      "other_stops_per_pop", "train_stat_per_1000", "PT_fact_medium",
                      "bus_stops_per_pop")]

# set row names according to canton labels
row.names(cl.an.cant.input) <- d.cant.norm.share$canton
```

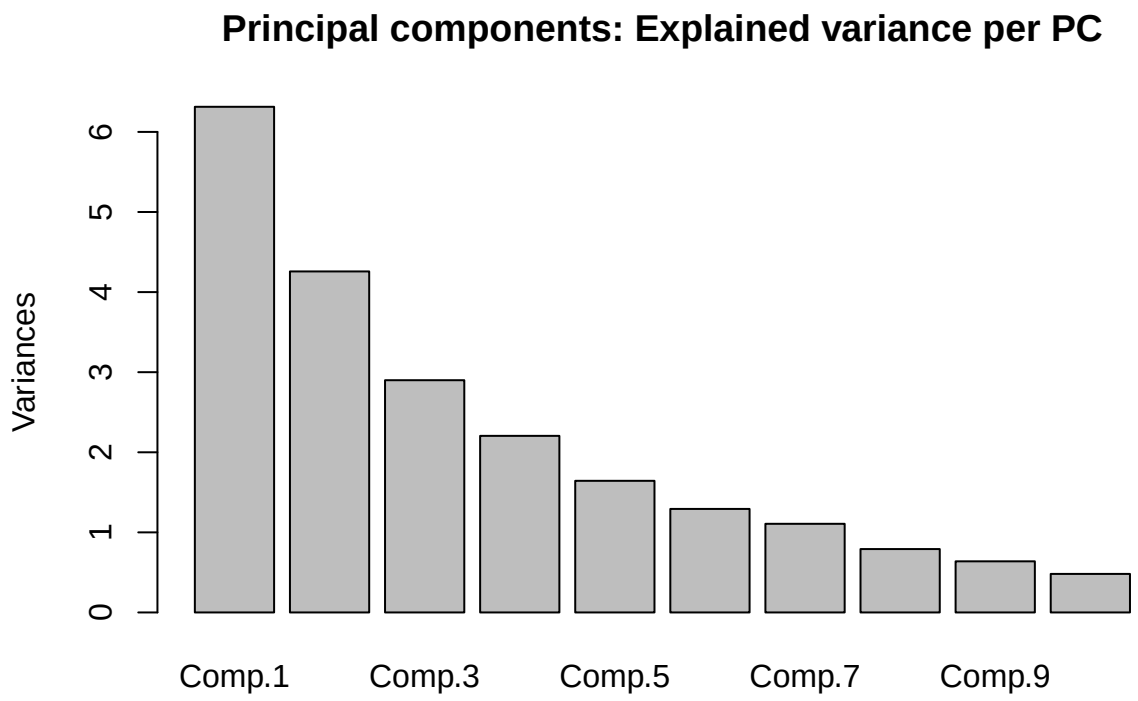
4.2.2 Preparation of clustering

```
# distance matrix
dp_cant <- dist(cl.an.cant.input)

# Principal components
PC.inf.fac.cant <- princomp(cl.an.cant.input) # without language
summary(PC.inf.fac.cant, loadings = FALSE)
```

```
## Importance of components:
##                           Comp.1    Comp.2    Comp.3     Comp.4     Comp.5
## Standard deviation     2.512686 2.0634228 1.7028587 1.48489202 1.28197927
## Proportion of Variance 0.273589 0.1845009 0.1256549 0.09554585 0.07121707
## Cumulative Proportion  0.273589 0.4580899 0.5837448 0.67929064 0.75050771
##                           Comp.6     Comp.7     Comp.8     Comp.9    Comp.10
## Standard deviation     1.13683275 1.05207005 0.88961478 0.79899706 0.69367111
## Proportion of Variance 0.05600351 0.04796356 0.03429463 0.02766384 0.02085112
## Cumulative Proportion  0.80651122 0.85447478 0.88876941 0.91643324 0.93728436
##                           Comp.11    Comp.12    Comp.13    Comp.14     Comp.15
## Standard deviation     0.62107403 0.55136949 0.54185768 0.37957123 0.307832714
## Proportion of Variance 0.01671509 0.01317369 0.01272309 0.00624322 0.004106309
## Cumulative Proportion  0.95399946 0.96717315 0.97989624 0.98613946 0.990245768
##                           Comp.16     Comp.17     Comp.18      Comp.19
## Standard deviation     0.289400362 0.241991072 0.228073617 0.1212260569
## Proportion of Variance 0.003629278 0.002537586 0.002254095 0.0006368161
## Cumulative Proportion  0.993875046 0.996412632 0.998666727 0.9993035430
##                            Comp.20      Comp.21      Comp.22      Comp.23
## Standard deviation     0.0958560874 0.0700329322 4.019855e-02 1.828023e-02
## Proportion of Variance 0.0003981635 0.0002125332 7.002335e-05 1.448056e-05
## Cumulative Proportion  0.9997017065 0.9999142397 9.999843e-01 9.999987e-01
```

```
##                             Comp.24
## Standard deviation     5.384632e-03
## Proportion of Variance 1.256418e-06
## Cumulative Proportion  1.000000e+00
```

```
## first 2 PC explain 46% of variance, 5 PC about 75% of variance

plot(PC.inf.fac.cant, main="Principal components: Explained variance per PC")
```

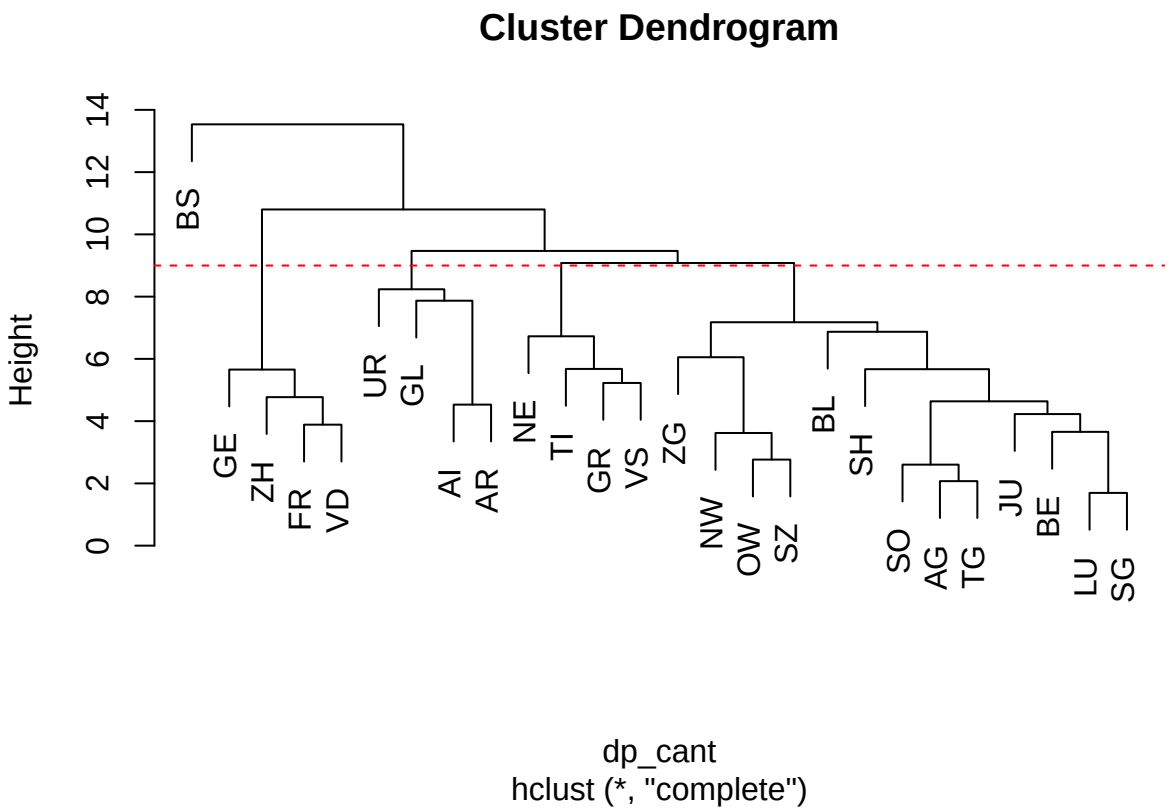



```
# write two first components out
pc.cant <- PC.inf.fac.cant$scores[,1:2]
```

4.2.3 Agglomerative Clustering

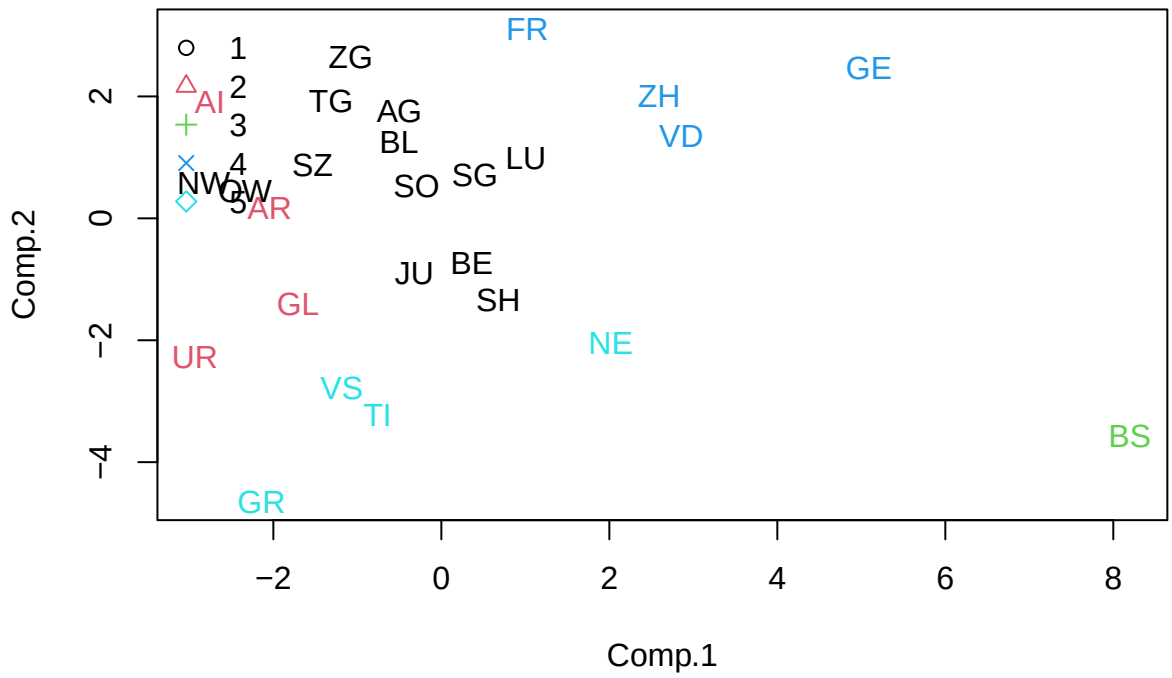
Again here, first the approahc of agglomerative clustering.

```
## Agglomerative clustering with complete linkage
cl.cant.inf.fac <- hclust(dp_cant, method = "complete")

## Show dendrogram
plot(cl.cant.inf.fac)
abline(h=9, col="red", lty=2)
```



Splitting at height 9 would result into 5 different clusters:

```
## Split into groups
k = 5 # split at height 9 => 5 groups
grps_cant <- cutree(cl.cant.inf.fac, k = k)


# plotting clustering result
plot(pc.cant, pch = grps_cant, col=grps_cant, lwd=2, type="n") # add clustering result
legend("topleft", legend = 1:k, pch = 1:k, col=1:k, bty="n")
text(pc.cant[,1], pc.cant[,2], labels=d.cant.norm.share$canton, col=grps_cant)
```




```
## Look at means of clusters / centroids
aggregate(cl.an.cant.input, by=list(cluster=grps_cant), mean)
```

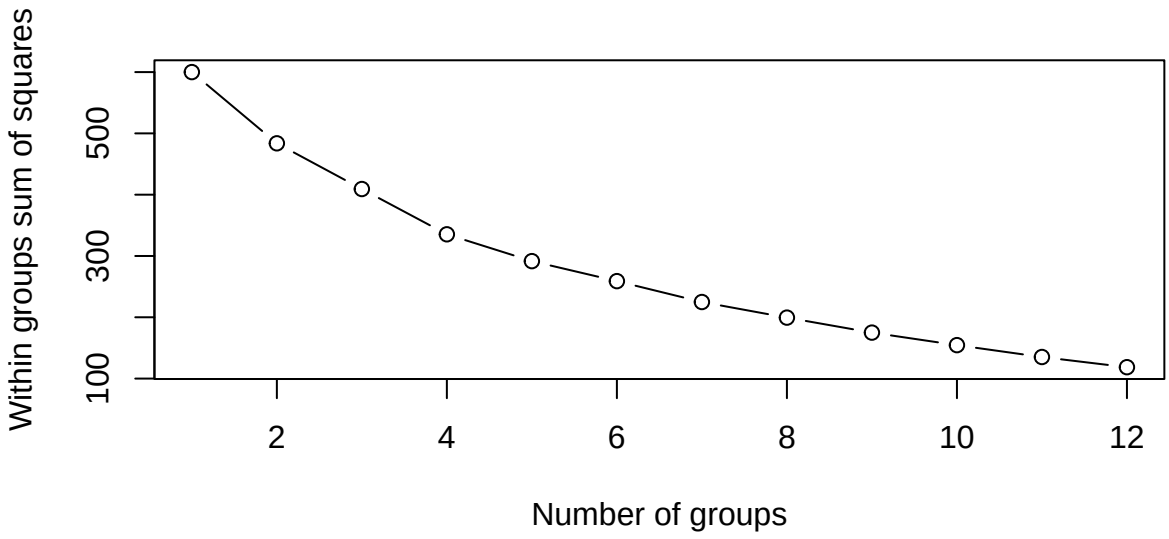
```
##   cluster age0_20_share age20_40_share birth_munic_share birth_cant_share
## 1       1     0.0138614     -0.1822481        -0.3682841       0.08717895
## 2       2     0.4144378     -0.2881687         0.6559767      -0.31908379
## 3       3    -1.8788670      2.2386642         3.2214163      -2.74798576
## 4       4     1.1861764      1.3004697        -0.5490215       0.25511454
## 5       5    -1.1759469     -0.9796607         0.2846139       0.46763410
##   birth_CH_share male_share resid_6_10y_share hh_1_share hh_2_share
## 1      0.3774966  0.3582343       -0.18810686 -0.3598379  0.5667078
## 2      0.7110705  0.9401891        0.97896326 -0.6653747  0.5967934
## 3     -0.7726282 -0.7144058       -0.27386238  3.2751432 -1.7734075
## 4     -0.8627021 -0.5274804       -0.09045127  0.0313340 -1.2050783
## 5     -0.8820755 -1.3983685       -0.20869909  0.9847279 -0.7901635
##   hh_3_5_share PT_time_medium PT_time_big train_stops_per_pop bus_stat_per_1000
## 1   -0.0404225    -0.13481799  -0.2200022          -0.2270643        -0.2378860
## 2    0.1957775     0.44008379   0.5086038           1.2067392         0.5583738
## 3   -2.8188930     1.81390328  -1.5713455          -1.6960204        -1.3552683
## 4    1.0909767    -0.36245031  -0.8592068          -0.5177394        -0.8285211
## 5   -0.4506579    -0.09295084   1.4584466           0.4729643         1.3820938
##   other_stat_per_1000 comb_car_per_1000 el_car_per_1000  PT_fact_big
## 1          -0.1582558         0.2739854       0.2746391  0.009474241
## 2           0.8671524         0.4013180      -0.6288553 -0.163032345
## 3          -0.7191758        -3.3862895      -1.1597302  1.240494269
```

```
## 4          -0.6355556        -0.8767527       0.2474265  0.999603035
## 5           0.4625286         0.4315545      -0.2212156 -1.177485539
##   single_share married_share other_stops_per_pop train_stat_per_1000
## 1   -0.4382117     0.4579125          -0.2081932          -0.3462886
## 2   -0.3065803     0.6702589           0.5703370           1.3805715
## 3    2.1289961    -2.5420811          -0.3901727          -1.4640645
## 4    1.5082327    -1.2237077          -0.5246842          -0.6039516
## 5   -0.3097136    -0.2992465           0.7285183           0.7148344
##   PT_fact_medium bus_stops_per_pop
## 1    -0.30068441       -0.13768471
## 2     0.43808046       -0.60295192
## 3    -2.12289204        2.34714880
## 4     1.16689793       -0.09192375
## 5    -0.09703105        0.55556376
```

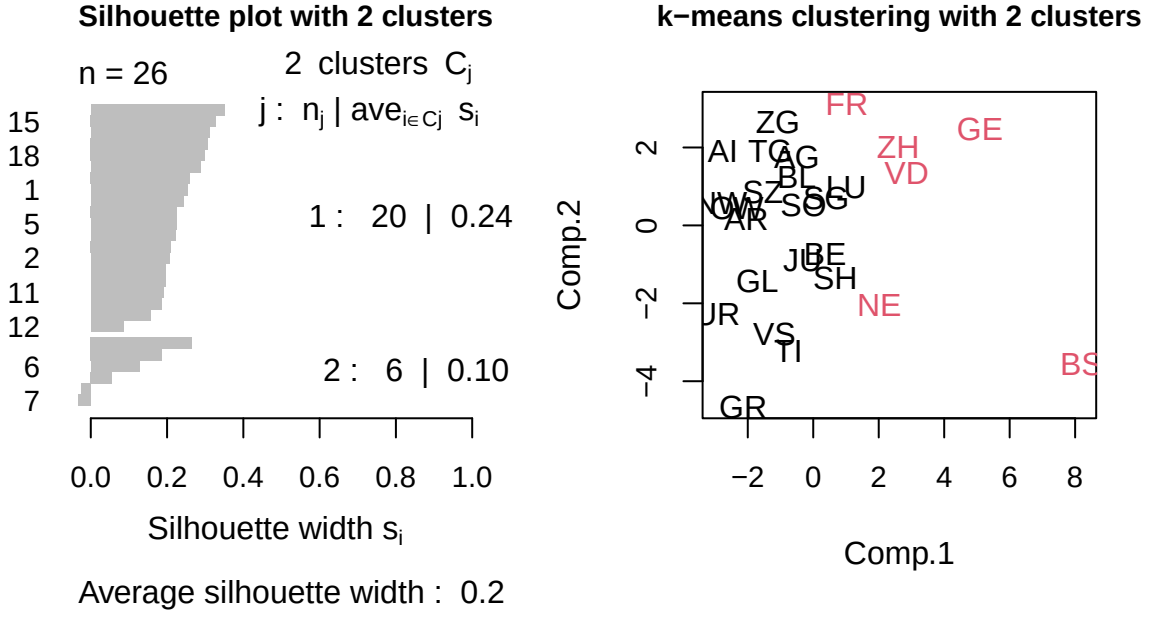
4.2.4 K-means clustering

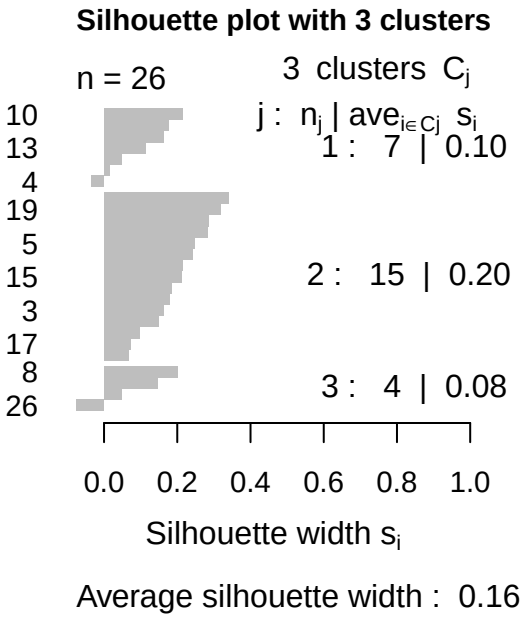
Now the second approahc, the k-means clustering

```
##############
## k-means ##
##############

## Choose number of clusters k with scree plot
nr <- 12
wss <- rep(0, nr)
for (i in 1:nr) wss[i] <- sum(kmeans(cl.an.cant.input, centers = i, nstart = 20)$withinss)
plot(1:nr, wss, type = "b", xlab = "Number of groups", ylab = "Within groups sum of squares")
```

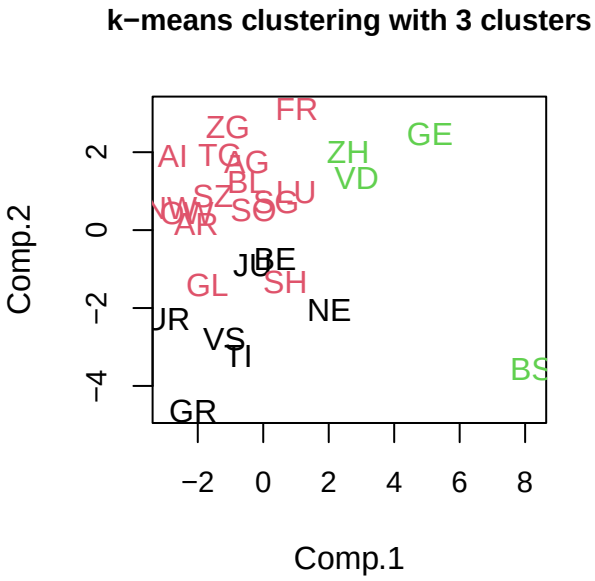


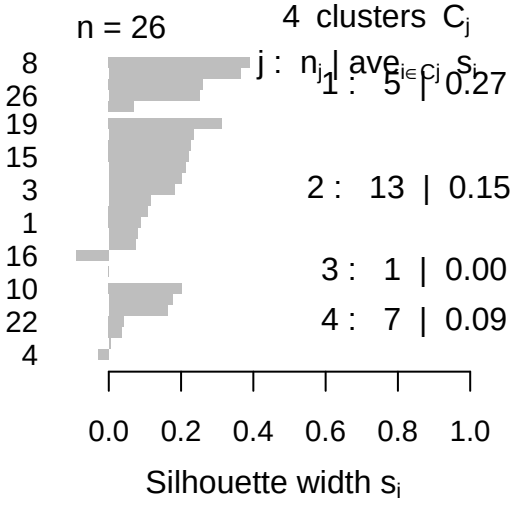
```r
## good spot at 4, test all from 2 to 10

par(mfrow=c(1,2), mar=c(5,4,4,2), cex.main = 0.9, cex.axis=0.9)
for(k in 2:10) {

  ## k-means with 3-10 centers
  ckm.cant <- kmeans(cl.an.cant.input, centers = k, nstart = 10)
  grpsKM.cant <- ckm.cant$cluster

  ## Silhouette Plot
  plot(silhouette(grpsKM.cant, dp_cant),
       main=paste("Silhouette plot with", k, "clusters"))

  ## visualize in PC 1 & 2
  plot(pc.cant, pch = grpsKM.cant, col=grpsKM.cant, lwd=2,
       main=paste("k-means clustering with", k, "clusters"), type="n")
  # legend("topleft", legend = 1:k, pch = 1:k, col=1:k, bty="n")
  text(pc.cant[,1], pc.cant[,2], labels=d.cant.norm.share$canton, col=grpsKM.cant)

}
```



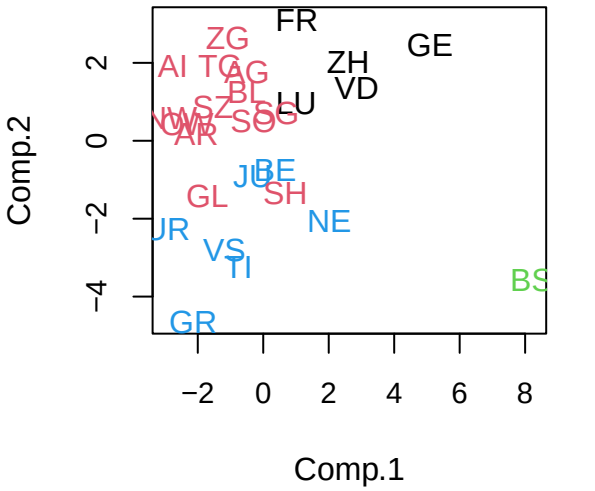


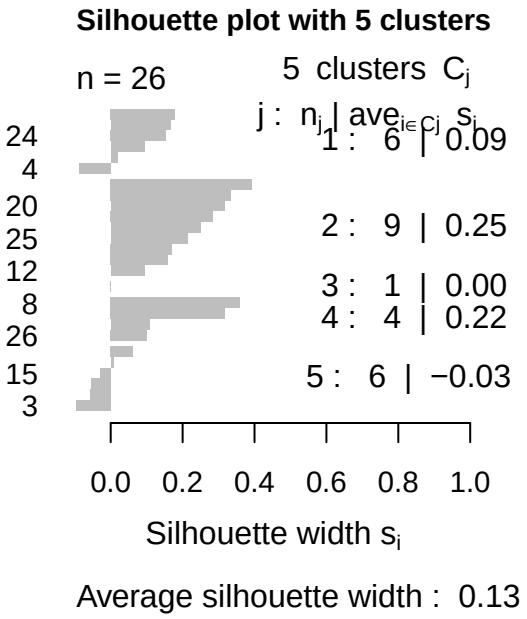
Average silhouette width : 0.16

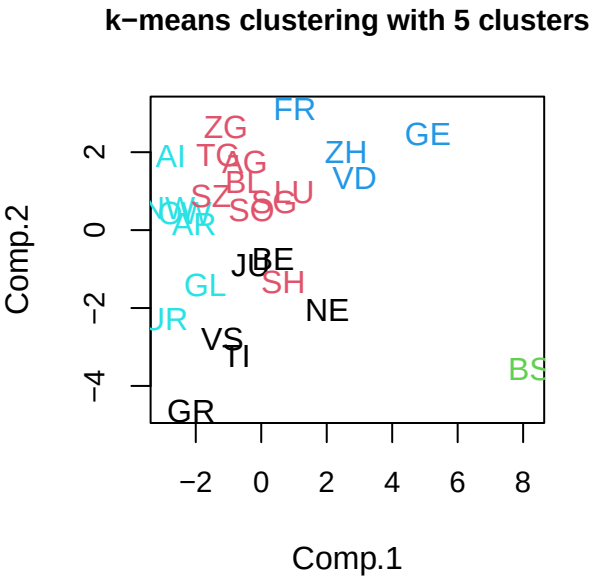


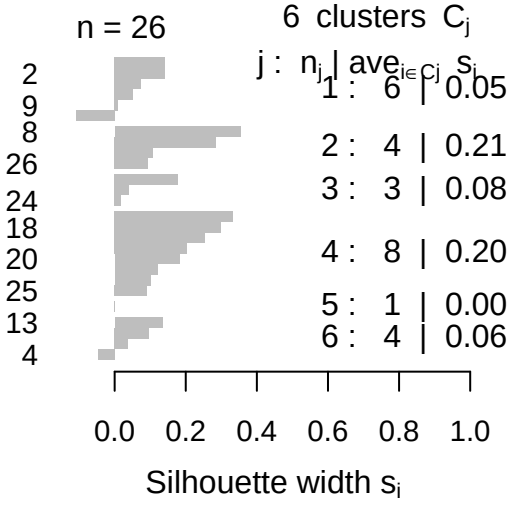


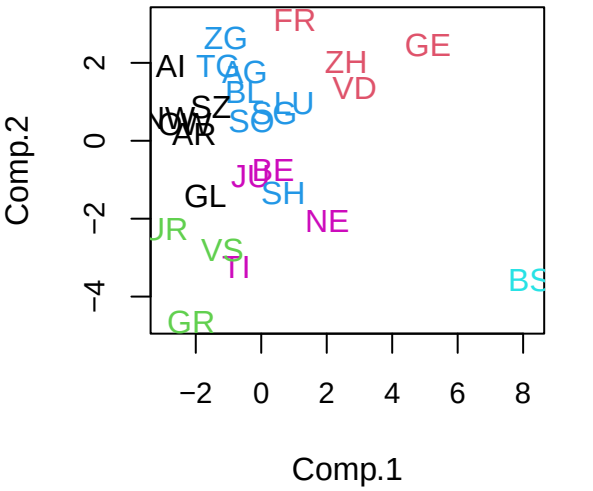
Average silhouette width : 0.15

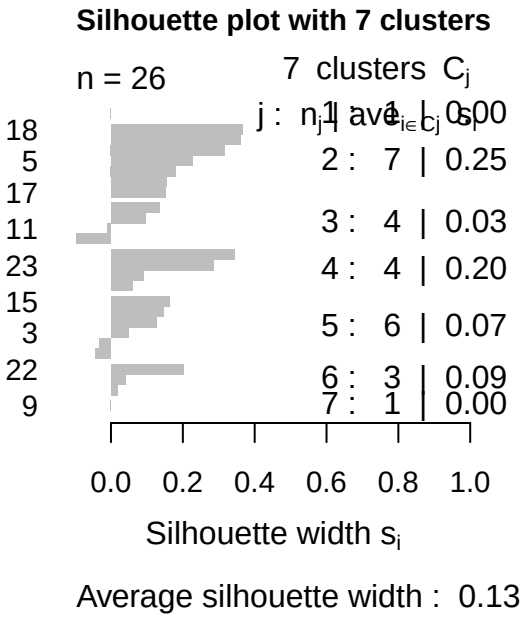


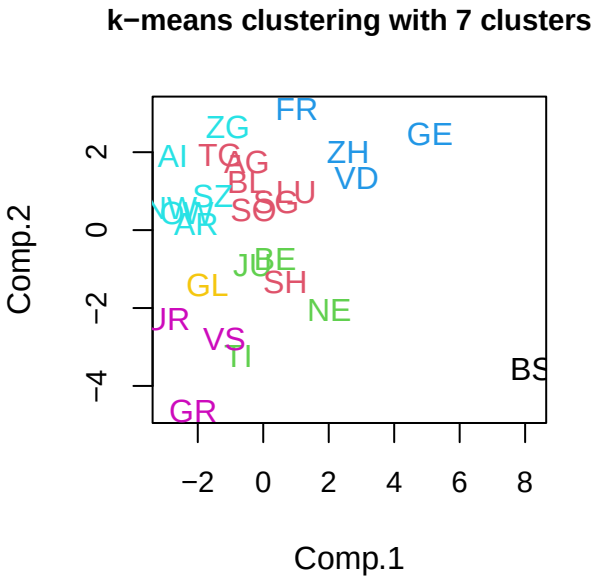

Silhouette plot with 5 clusters
n = 26
5 clusters C_j
j : n_j | $ave_{i \in Cj}$ s_i
1 : 6 | 0.09
2 : 9 | 0.25
3 : 1 | 0.00
4 : 4 | 0.22
5 : 6 | −0.03
24
4
20
25
12
8
26
15
3
0.0 0.2 0.4 0.6 0.8 1.0
Silhouette width s_i
Average silhouette width : 0.13

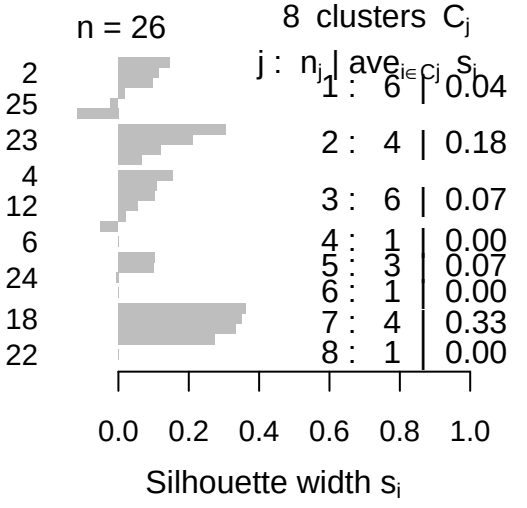

k−means clustering with 5 clusters
Comp.2
Comp.1
FR
GE
ZG
AI
TG
AG
ZH
VD
BL
SZ
SH
LU
SO
SG
AR
JU
BE
GL
SH
NE
JR
VS
TI
BS
GR
−2 0 2 4 6 8
−4 −2 0 2

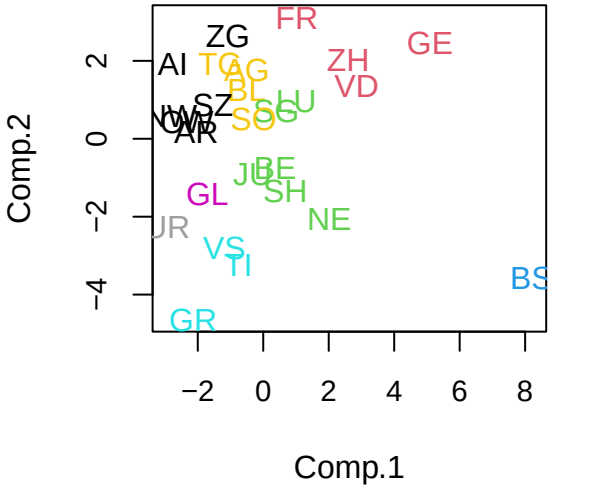

Silhouette plot with 6 clusters
n = 26
6 clusters C_j
j : n_j | $ave_{i \in Cj}$ s_i
1 : 6 | 0.05
2 : 4 | 0.21
3 : 3 | 0.08
4 : 8 | 0.20
5 : 1 | 0.00
6 : 4 | 0.06
2
9
8
26
24
18
20
25
13
4
0.0 0.2 0.4 0.6 0.8 1.0
Silhouette width s_i
Average silhouette width : 0.12

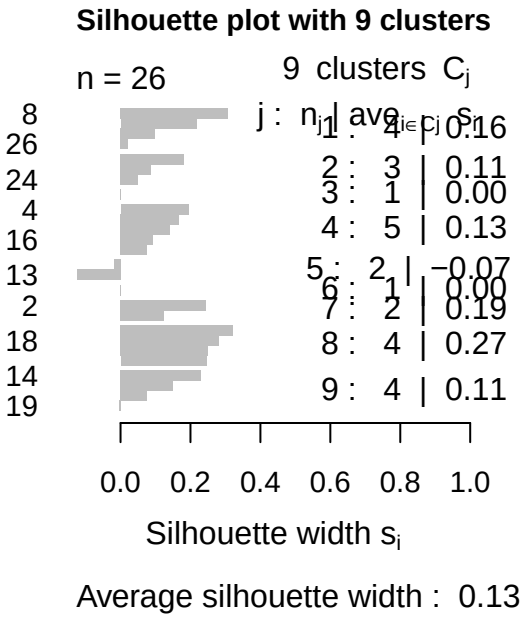

k−means clustering with 6 clusters
Comp.2
Comp.1
FR
GE
ZG
AI
TG
AG
ZH
VD
BL
SZ
SH
LU
SO
SG
AR
JU
BE
GL
SH
NE
JR
VS
TI
BS
GR
−2 0 2 4 6 8
−4 −2 0 2

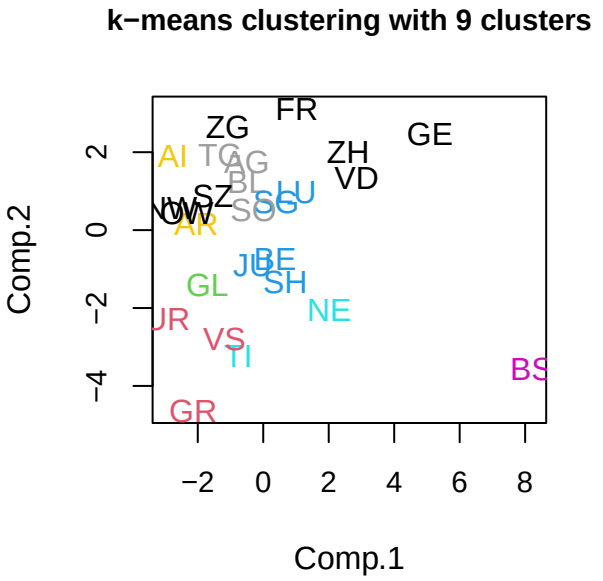

Silhouette plot with 7 clusters
n = 26
7 clusters C_j
j : n_j | ave_{i∈Cj} s_i
1 : 4 | 0.00
2 : 7 | 0.25
3 : 4 | 0.03
4 : 4 | 0.20
5 : 6 | 0.07
6 : 3 | 0.09
7 : 1 | 0.00
18
5
17
11
23
15
3
22
9
0.0 0.2 0.4 0.6 0.8 1.0
Silhouette width s_i
Average silhouette width : 0.13

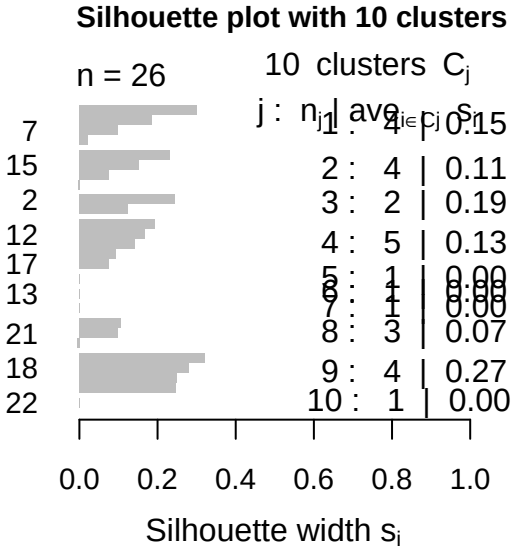

k−means clustering with 7 clusters
Comp.2
Comp.1
FR
GE
ZG
AI
TG
ZH
VD
BL
SZ
SO
AR
JU
BE
SH
GL
NE
JR
VS
TI
BS
GR
−4 −2 0 2
−2 0 2 4 6 8

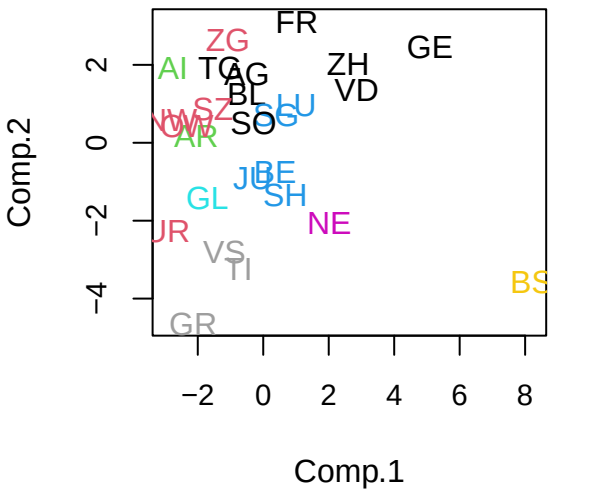

Silhouette plot with 8 clusters
n = 26
8 clusters C_j
j : n_j | ave_{i∈Cj} s_i
1 : 6 | 0.04
2 : 4 | 0.18
3 : 6 | 0.07
4 : 1 | 0.00
5 : 3 | 0.07
6 : 1 | 0.00
7 : 4 | 0.33
8 : 1 | 0.00
2
25
23
4
12
6
24
18
22
0.0 0.2 0.4 0.6 0.8 1.0
Silhouette width s_i
Average silhouette width : 0.11


k−means clustering with 8 clusters
Comp.2
Comp.1
FR
GE
ZG
AI
TG
ZH
VD
BL
SZ
SO
AR
JU
BE
SH
GL
NE
JR
VS
TI
BS
GR
−4 −2 0 2
−2 0 2 4 6 8







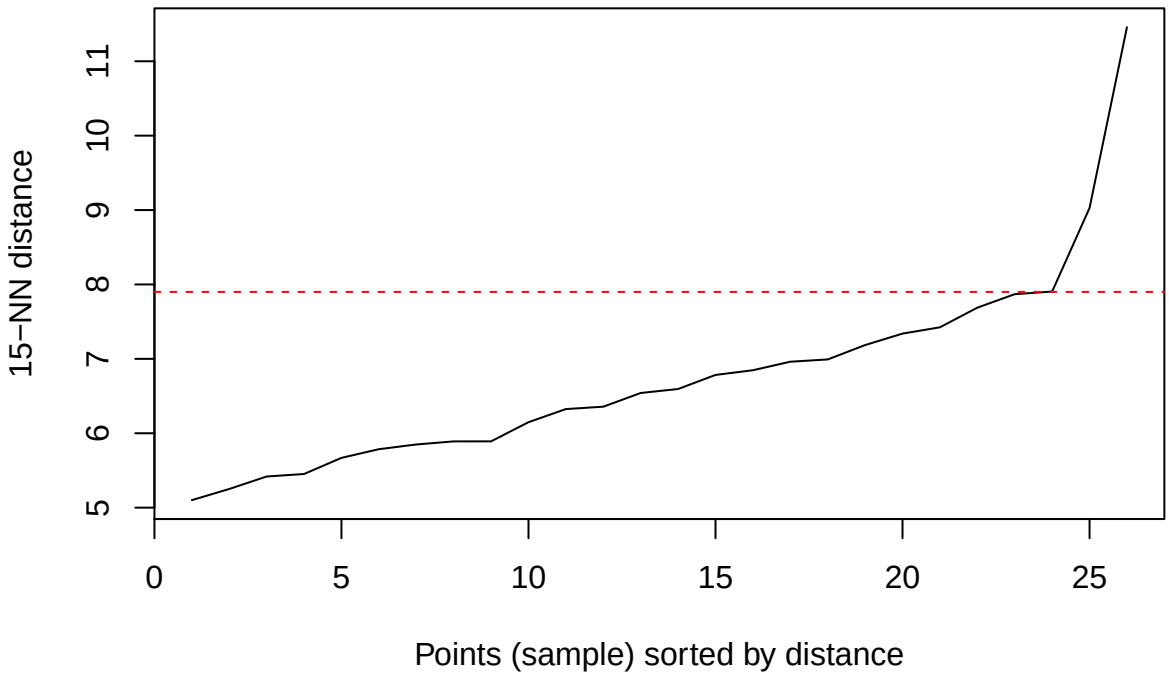



```
### 4 clusters have the highest average silhouette width!

# k-means cluster with k=4
ckm4.cant <- kmeans(cl.an.cant.input, centers = 4, nstart = 10)
grpsKM4.cant <- ckm4.cant$cluster
```

Somewhere around 4 clusters seems to be accurate. Going from the silhouette value, I will take 8 as value.

4.2.5 DBSCAN

```
#############
## DBSCAN  ##
#############

kNNdistplot(cl.an.cant.input, k = 15) ## -> eps value = 7.9 (sharp increase)
abline(h=7.9, col = "red", lty=2)
```




```
# setting parameters according to minPts = k and eps = sharp increase

for(i in (1:7)) {

  for(j in (1:4)) {

    dbs.cant <- dbscan(cl.an.cant.input, eps = i, minPts = j)

    print(paste("minPts:", j, "; eps: ", i, "; nr. of clusters: ",
                max(dbs.cant$cluster))) # 1
  }

}
```

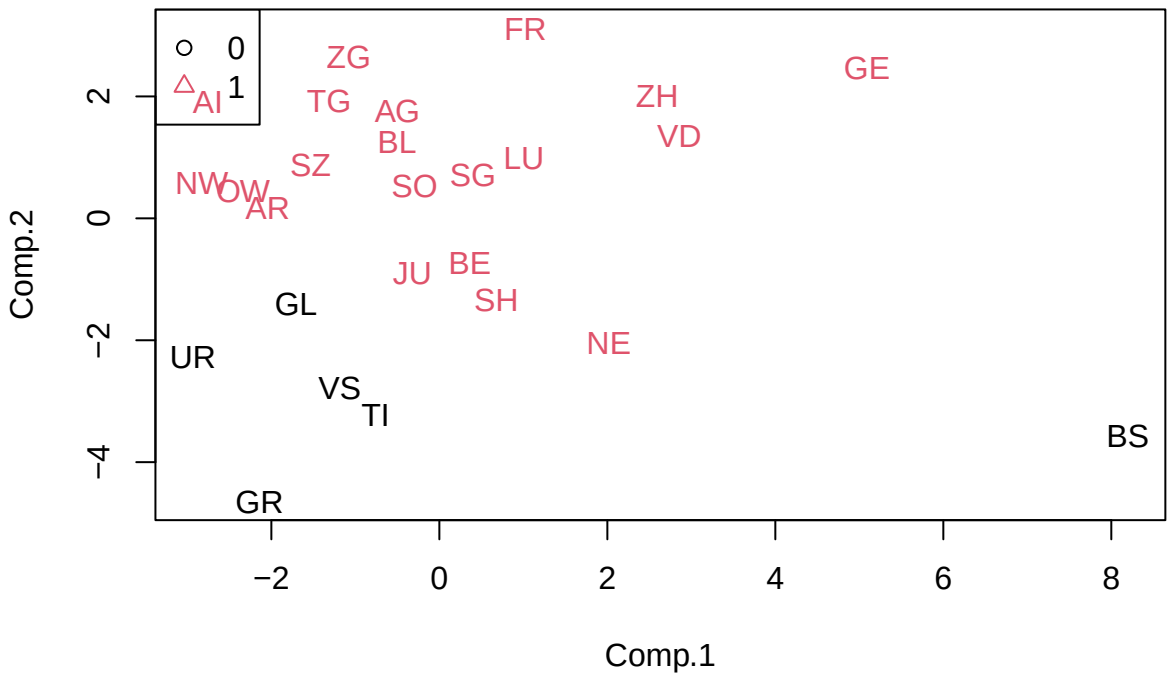
```
## [1] "minPts: 1 ; eps:  1 ; nr. of clusters:  26"
```

```
## [1] "minPts: 2 ; eps:  1 ; nr. of clusters:  0"
## [1] "minPts: 3 ; eps:  1 ; nr. of clusters:  0"
## [1] "minPts: 4 ; eps:  1 ; nr. of clusters:  0"
## [1] "minPts: 1 ; eps:  2 ; nr. of clusters:  25"
## [1] "minPts: 2 ; eps:  2 ; nr. of clusters:  1"
## [1] "minPts: 3 ; eps:  2 ; nr. of clusters:  0"
## [1] "minPts: 4 ; eps:  2 ; nr. of clusters:  0"
## [1] "minPts: 1 ; eps:  3 ; nr. of clusters:  20"
## [1] "minPts: 2 ; eps:  3 ; nr. of clusters:  1"
## [1] "minPts: 3 ; eps:  3 ; nr. of clusters:  1"
## [1] "minPts: 4 ; eps:  3 ; nr. of clusters:  1"
## [1] "minPts: 1 ; eps:  4 ; nr. of clusters:  12"
## [1] "minPts: 2 ; eps:  4 ; nr. of clusters:  1"
## [1] "minPts: 3 ; eps:  4 ; nr. of clusters:  1"
## [1] "minPts: 4 ; eps:  4 ; nr. of clusters:  1"
## [1] "minPts: 1 ; eps:  5 ; nr. of clusters:  7"
## [1] "minPts: 2 ; eps:  5 ; nr. of clusters:  1"
## [1] "minPts: 3 ; eps:  5 ; nr. of clusters:  1"
## [1] "minPts: 4 ; eps:  5 ; nr. of clusters:  1"
## [1] "minPts: 1 ; eps:  6 ; nr. of clusters:  3"
## [1] "minPts: 2 ; eps:  6 ; nr. of clusters:  1"
## [1] "minPts: 3 ; eps:  6 ; nr. of clusters:  1"
## [1] "minPts: 4 ; eps:  6 ; nr. of clusters:  1"
## [1] "minPts: 1 ; eps:  7 ; nr. of clusters:  2"
## [1] "minPts: 2 ; eps:  7 ; nr. of clusters:  1"
## [1] "minPts: 3 ; eps:  7 ; nr. of clusters:  1"
## [1] "minPts: 4 ; eps:  7 ; nr. of clusters:  1"
```

```
# minPts=1 and eps=5 lead to 7 clusters, eps=6 to 3 clusters. Test these!

minPts=2
eps=5

dbs.cant <- dbscan(cl.an.cant.input, eps = eps, minPts = minPts)

max(dbs.cant$cluster) # 7, look at values
```

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

```
## plot on PC 1 & 2
plot(pc.cant, pch = dbs.cant$cluster+1, col = dbs.cant$cluster+1, type="n")
legend("topleft", legend = 0:max(dbs.cant$cluster), pch = 1:(max(dbs.cant$cluster)+1),
       col = 1:(max(dbs.cant$cluster)+1))
text(pc.cant[,1], pc.cant[,2], labels=d.cant.norm.share$canton, col=dbs.cant$cluster+1)
```




```
table(dbs.cant$cluster)
```

```
##
##  0  1
##  6 20
```

It is always the case that either only one cluster is created, or individual cantons then form their own cluster. It can be concluded from this that no suitable cluster analysis is possible with DBSCAN.

4.2.6 Gaussian mixture model

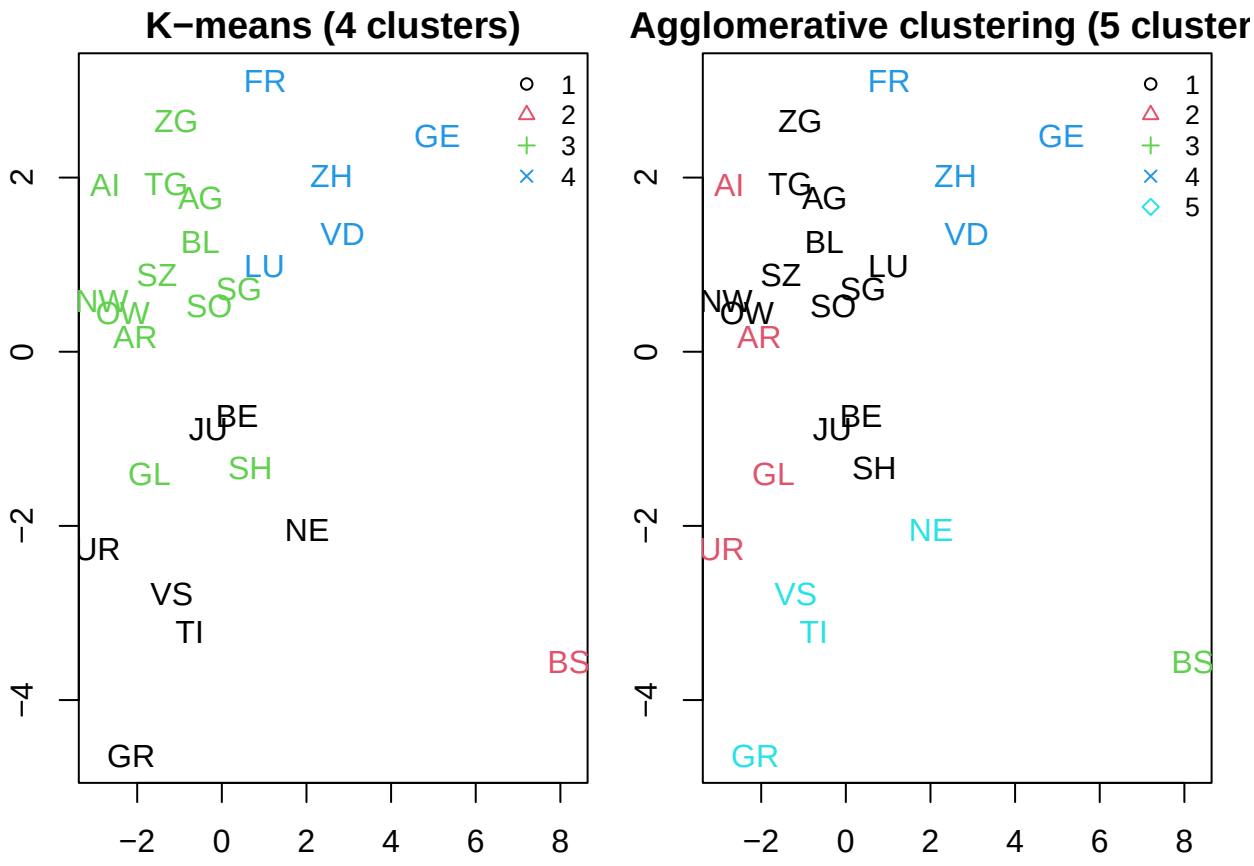
```
## Gaussian Mixture Models

mc.cant <- Mclust(cl.an.cant.input)
table(mc.cant$classification)
```

```
##
##  1  2
## 25  1
```

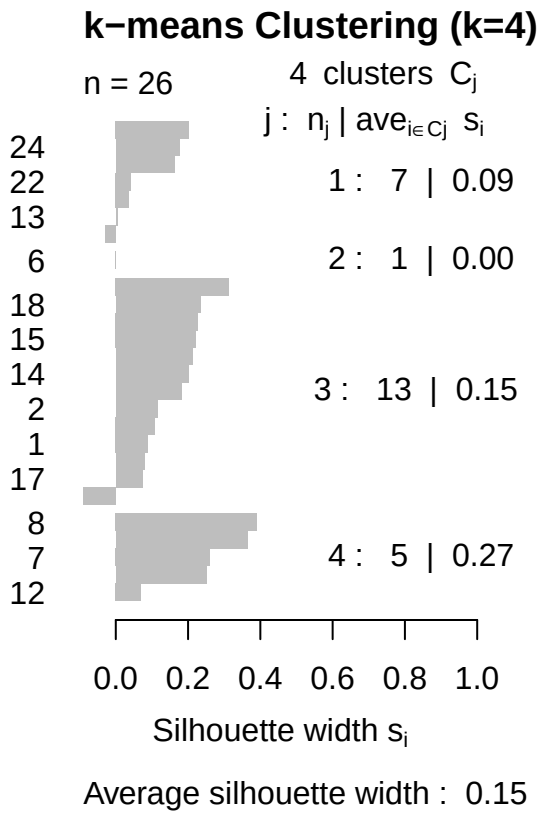
This two classes built are not very useful.

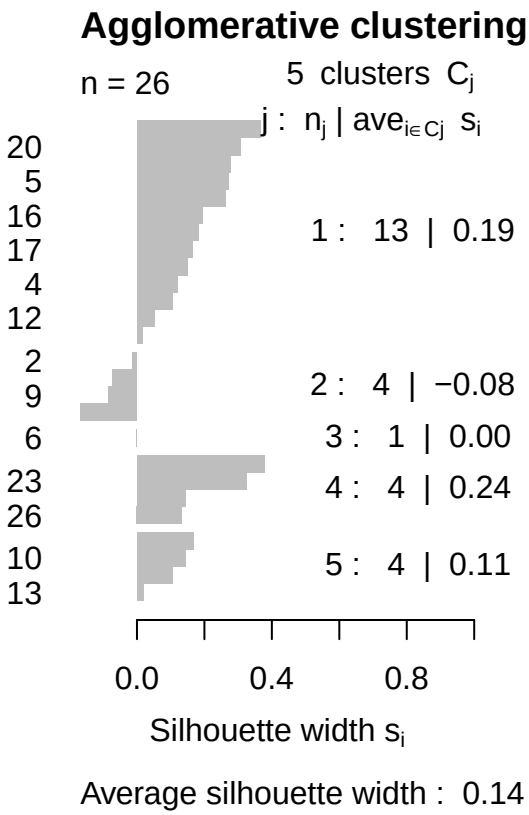
4.2.7 Comparison of different approaches

```
par(mfrow=c(1,2), mar=c(2,2,1.5,1))

# K-means with 4 clusters
plot(pc.cant, pch = grpsKM4.cant, col=grpsKM4.cant, main="K-means (4 clusters)", type="n")
legend("topright", legend = 1:4, pch = 1:4, col=1:4, bty="n", cex=0.8)
text(pc.cant[,1], pc.cant[,2], labels=d.cant.norm.share$canton, col=grpsKM4.cant)

# Agglomerative clustering
plot(pc.cant, pch = grps_cant, col=grps_cant,
     main="Agglomerative clustering (5 clusters)", type="n") # add clustering result
legend("topright", legend = 1:5, pch = 1:5, col=1:5, bty="n", cex=0.8)
text(pc.cant[,1], pc.cant[,2], labels=d.cant.norm.share$canton, col=grps_cant)
```




```
## Silhouette values ##

# Silhouette plot from package "cluster"
par(mfrow=c(1,2), mar=c(5,4,4,2))

plot(silhouette(grpsKM4.cant, dp_cant), main="k-means Clustering (k=4)")
plot(silhouette(grps_cant, dp_cant), main="Agglomerative clustering")
```





The k-means clustering looks slightly better, with no group having a negative average silhouette width!

4.2.8 Exporting enriched data

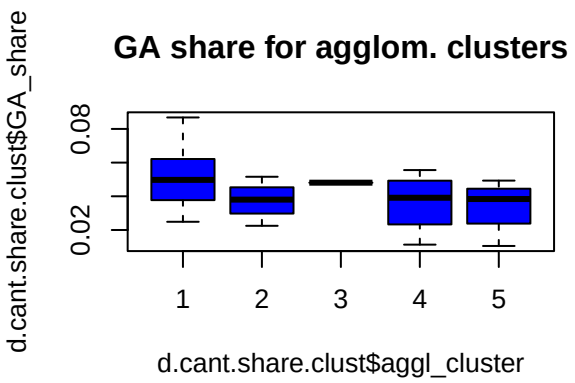
The generated clusters can now be written into a table together with the relevant influencing factors, which can then be used for the distribution of shares and graphical analysis with Tableau.

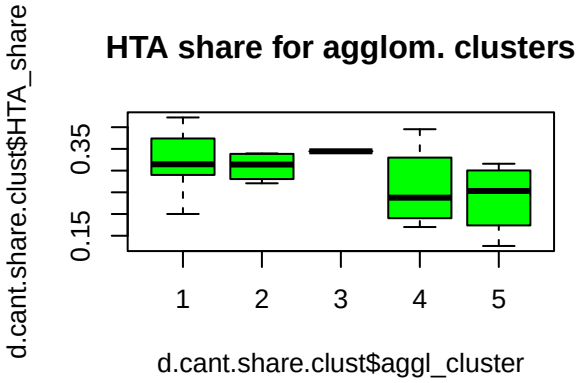
```
d.cant.share.clust <- d.cant.share
d.cant.share.clust["k_cluster"] = grpsKM4.cant
d.cant.share.clust["aggl_cluster"] = grps_cant

# categorical data
d.cant.share.clust$k_cluster<-as.factor(d.cant.share.clust$k_cluster)
d.cant.share.clust$aggl_cluster<-as.factor(d.cant.share.clust$aggl_cluster)

write.csv(d.cant.share.clust, file="../Data/4_Model_output/cant_influence_factors_with_CA.csv")
```

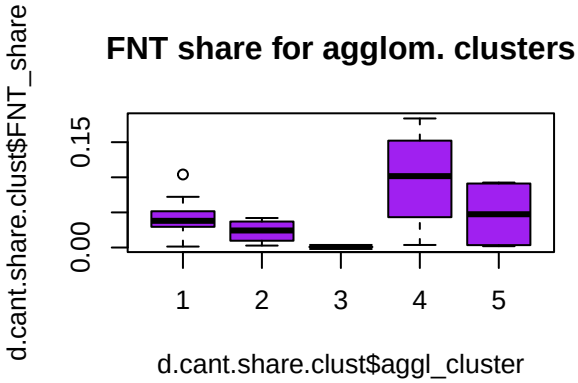
4.2.9 Share distribution of GA, HTA and FNT for clusters

```
par(mfrow=c(2,2))
boxplot(d.cant.share.clust$GA_share ~ d.cant.share.clust$aggl_cluster, col="blue",
        main = "GA share for agglom. clusters")
boxplot(d.cant.share.clust$HTA_share ~ d.cant.share.clust$aggl_cluster, col="green",
        main = "HTA share for agglom. clusters")
```

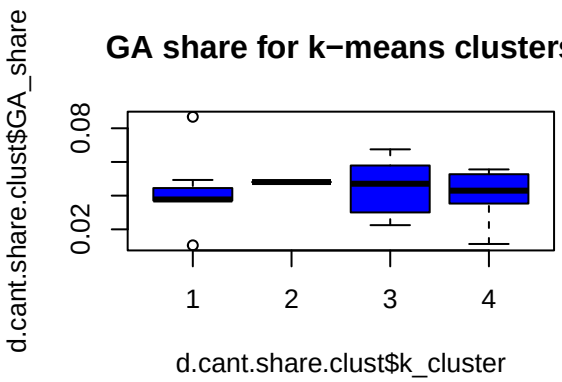
```
boxplot(d.cant.share.clust$FNT_share ~ d.cant.share.clust$aggl_cluster, col="purple",
        main = "FNT share for agglom. clusters")
```

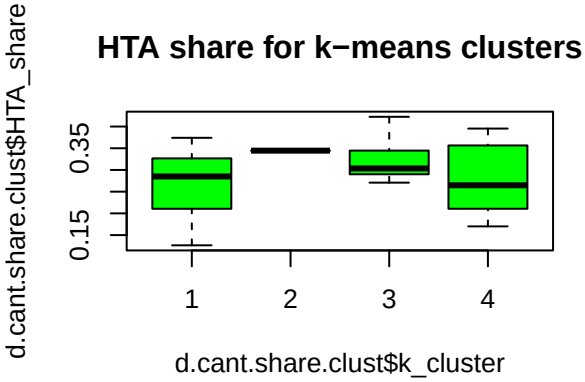


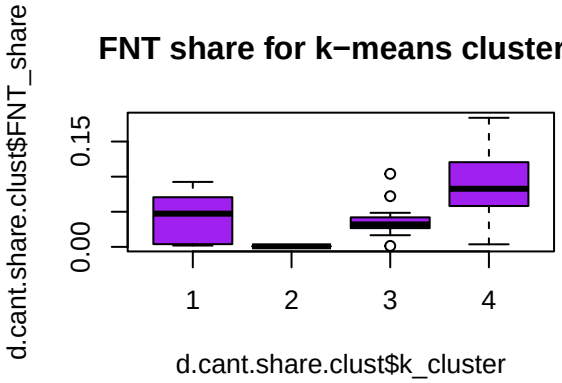





```
par(mfrow=c(2,2))
boxplot(d.cant.share.clust$GA_share ~ d.cant.share.clust$k_cluster, col="blue",
        main = "GA share for k-means clusters")
boxplot(d.cant.share.clust$HTA_share ~ d.cant.share.clust$k_cluster, col="green",
        main = "HTA share for k-means clusters")
boxplot(d.cant.share.clust$FNT_share ~ d.cant.share.clust$k_cluster, col="purple",
        main = "FNT share for k-means clusters")
```


GA share for k−means clusters
d.cant.share.clust$GA_share
0.02
0.08
1
2
3
4
d.cant.share.clust$k_cluster


HTA share for k−means clusters
d.cant.share.clust$HTA_share
0.15
0.35
1
2
3
4
d.cant.share.clust$k_cluster


FNT share for k−means clusters
d.cant.share.clust$FNT_share
0.00
0.15
1
2
3
4
d.cant.share.clust$k_cluster