CS5608 Big Data Analytics Coursework for 2018/19 21.04.2019

“Swiss referendum on TV tax analysis”

Introduction

On March the 4th 2018, in Switzerland, a people's vote took place on the “TV licence” tax abolition. The goal of this referendum was to abolish this tax mainly for liberal reasons. There has been very heated discussions around it, including social media. Polls showed that this idea was convincing a significant part of the population (about 40%). However this referendum was rejected by a large margin, as 71,6% of the population voted for “no”, same at cantons level: all rejected it (a double majority was needed for it to be accepted).

How can this vote be explained, looking at socio-economics parameters? Or political ones? If it can, what parameter would play the largest role in this explanation? Can we spot the famous “Rösti Graben” i.e. clear different response to political questions between French speaking cantons and German speaking ones? This a key question, since a canton majority is needed to accept a referendum, on top of a majority of voters overall. As about three quarters of cantons are German speaking, the others complain that their voice is never heard. Or is there rather a gap between large cities and small towns, which is the reason usually given to explain that the Rösti Graben is a myth and does not exist? We will to try to use machine learning methods to bring clarity into those questions, with data available.

Here we can underline that we will not try to predict whether an individual will vote “yes” or “no”, but how a group of people, at locality level, will be voting, as a numerical proportion of “yes” or “no”. This is a regression problem, in terms of machine learning problem. We will use both supervised and unsupervised techniques to try and make light in the available data and draw sensible conclusions based on it.

1. Data description and research question

Precisely, here are our research questions:

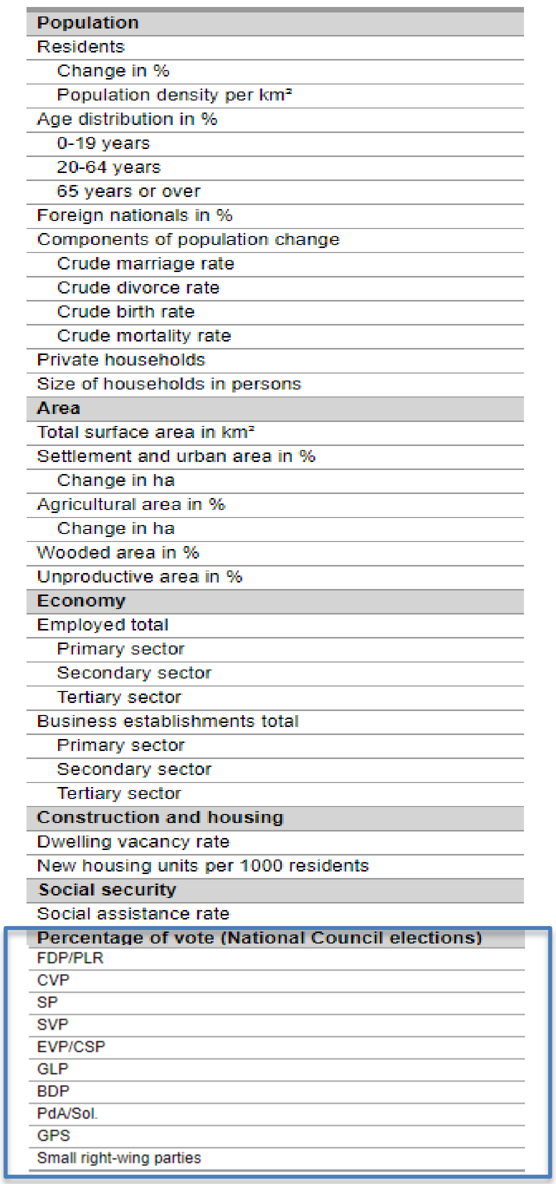
1) Do socio-economics, political affiliation or speaking language factors at locality level influence the referendum results?
2) If yes, which are the most important ones and to what extent would they globally enable to predict a result in a given locality?
3) Is there such a thing as the “Rösti Graben”, i.e. a cultural gap between French and German speaking citizens that is illustrated in the referendum results? Or can we rather observe a gap between large cities and small towns?

The referendum results per localities (towns) are public, so are localities socio-economics and political data. A list of official main language for each locality can be found too.

Referendum results and localities dataset s

The referendum dataset provides results about referendum per locality: 2227 localities, including “virtual” localities for Swiss nationals voting from abroad.

Column name	**Content / meaning**
No canton	Numerical ID of canton (1 to 26)
Canton	Name of the canton
No commune	Numerical ID of locality (official)
Commune	Name of the locality (in its official language)
Total voters	Number of people who took part in the vote
Total ballots	Total number of received ballots
Participation	Proportion of voters of total population
Blank ballots	Number of blank ballots
Invalid ballots	Number of invalid ballot
Valid ballots	Number of valid ballots
Yes	Number of yes
No	Number of no
Yes in %	Yes in proportion, i.e. number of yes over number of valid ballots



Population
- Residents
  - Change in %
  - Population density per km²
- Age distribution in %
  - 0-19 years
  - 20-64 years
  - 65 years or over
- Foreign nationals in %
- Components of population change
  - Crude marriage rate
  - Crude divorce rate
  - Crude birth rate
  - Crude mortality rate
- Private households
- Size of households in persons

Area
- Total surface area in km²
- Settlement and urban area in %
  - Change in ha
- Agricultural area in %
  - Change in ha
- Wooded area in %
- Unproductive area in %

Economy
- Employed total
  - Primary sector
  - Secondary sector
  - Tertiary sector
- Business establishments total
  - Primary sector
  - Secondary sector
  - Tertiary sector

Construction and housing
- Dwelling vacancy rate
- New housing units per 1000 residents

Social security
- Social assistance rate

Percentage of vote (National Council elections)
- FDP/PLR
- CVP
- SP
- SVP
- EVP/CSP
- GLP
- BDP
- PdA/Sol.
- GPS
- Small right-wing parties

Table above: referendum results data
On the right: localities portrait data

From this list we can already see that some information is redundant, i.e. some of the information can be directly deduced from other pieces of information.

The portrait dataset provides quite a detailed view of 2324 localities, in terms of socio-economics and politics aspects. As columns are self-explaining we won't comment them further.

The last part (blue box) is less obvious since it refers to Swiss political parties. The only party worth mentioning the "SVP", i.e. the Swiss People's Party is a national-conservative and right-wing populist political party. It is the largest party in Switzerland and was the only party to call for accepting this referendum. This piece of information is sent to all people who have a right to vote, in an informative leaflet edited by the government.

Official political parties voting advice for TV tax abolition (for information, no dataset)

Political party	**PLR**	**PDC**	**PS**	**SVP**	**PEV**	**PVL**	**PBD**	**PES**	**MCR**
Tax abolition	No	No	No	**Yes**	No	No	No	No	-

In total, this is about the file contains more than 40 features, with sum redundancy as well and some information we are not interested in, as we will see later.

List of official main language per locality

Geographic language repartition is Switzerland is rather clear: the east speaks German, the south of the Alps Italian and the rest French. Only a handful of localities speak “Romansh”. It even clearer at canton level: most cantons speak only one language apart from a few exceptions that are bilingual and these cantons actual speak mainly one language with a few communal exceptions.
A list has been found that delivers this information at the right granularity (i.e. locality). But it is PDF format and although R has got a library to extract data in this format, we will populate it by script as it seems like the most pragmatic approach.

2. Data preparation and cleaning

First step is to merge the 2 datasets, doing an “inner join” as we are only interested in complete data. The common key is the locality number. Left out are localities for which we have no portrait but voting results, i.e. Swiss from abroad, it is ok to not care about them.
Then we enrich with the language feature, based on canton list of localities exceptions. We rename columns where R has done some automatic characters replacement. At this stage, we can remove some columns that are twice in the dataset or that convey information we don’t need in our analysis for sure. As well, we remove the most obvious redundant columns, for example number of “No” when we have total number of valid votes and number of “Yes”.
Then we perform a few sanity checks, to make sure no data was forgotten in the process, no data is in double and no empty data (N/A) is to be reported.
We note that no data is a factor; features are mostly numerical, except locality names, canton names, and language indication.

Our complete dataset has got 2200 observations and 48 variables, no missing values, no duplicate values.

3. Exploratory data analysis

Data distribution

The first striking element to note is the number of localities, compared to the total population in the country. There are over 2’000 of them, for a population of about 8 million people,

which sets the average number of residents at about 3’680, whereas the median number is 1453.
Indeed, inspecting the data, we notice that several features have an imbalanced distribution, quite skewed to the left that reflects the fact that Swiss localities are of small size in majority. Only a few are very large, only 6 are above 100K: one at 400K (Zürich), one at 200K (Geneva) and the others closer to 100K (Bern, Basel and Lausanne).

Several algorithms account for a fairly normal data distribution, or, at least, will suffer from imbalances. Typically, it will be the case when using Euclidean distance or when trying to display the data. As well, performance of the k-means algorithm tends to be negatively affected by skewed data distributions. PCA is also impacted by outliers.
This issue affects two types of data: population numbers (number employees by sector, number of households,...) and area size (total surface). To remedy this, in cases where possible, we will turn these numbers from absolute numbers into percentages. Where not possible, we will apply a log function to the numbers. We graph these features to make sure the log transformation is making the data more balanced.

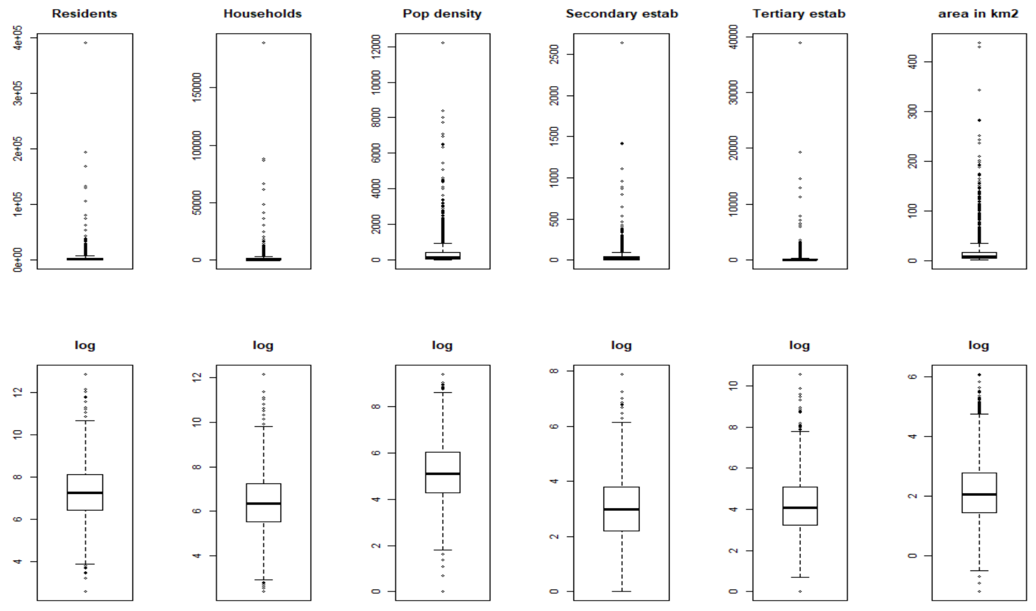
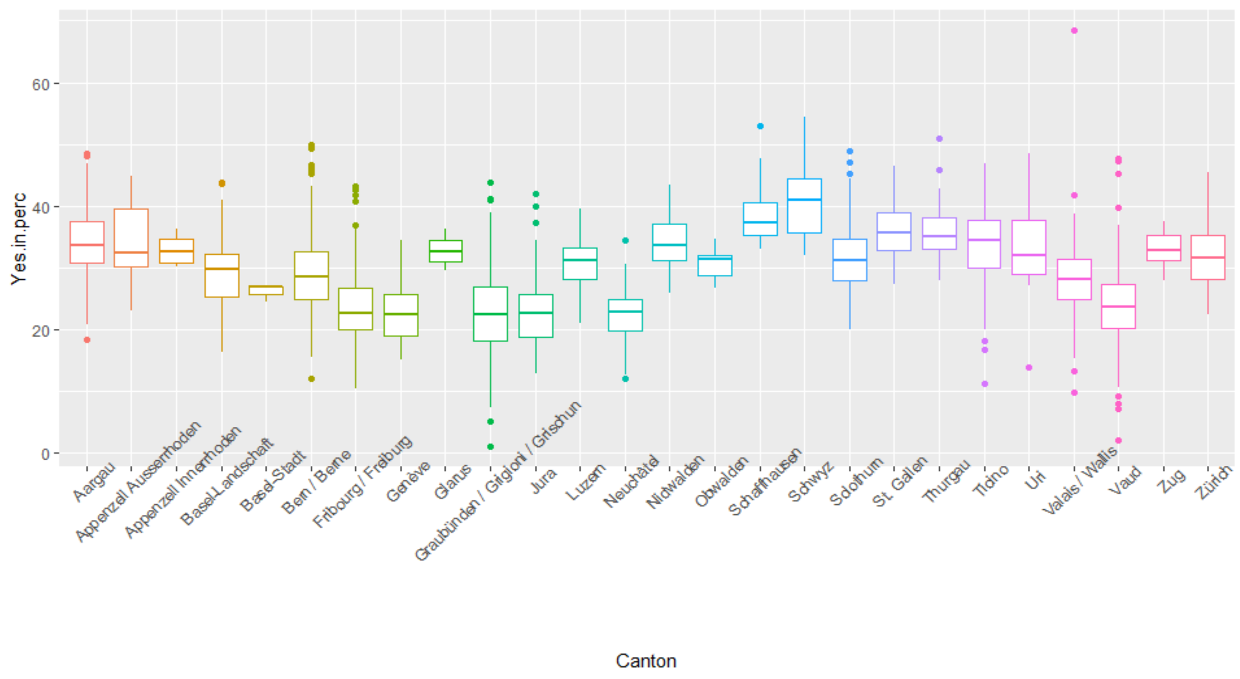
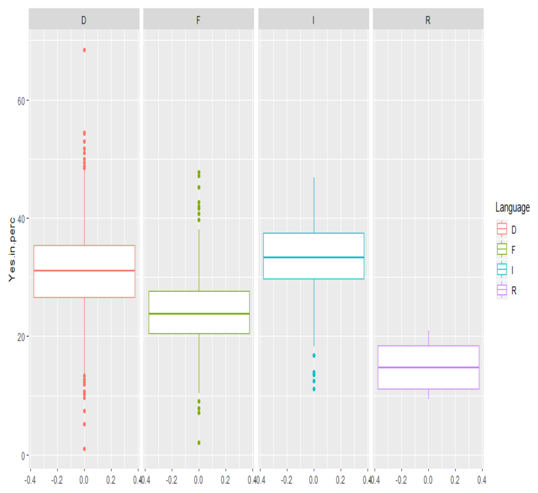


Fig 1. Features with skewed distributions, before and after log transformation

After checking the results, we see a much more balanced distribution, as expected. We will just need to “exponentiate” the values when we need to interpret them. Another option would have been to use discrete ordinal values, such as “large”, “medium”, “small”. Overall, this would have successfully solved the problem, but to the loss of potentially useful information.



Trying to get a flavour of the referendum results, one simple way to look at it is to plot the results by canton, as shown above. Interesting is to note the relatively important variability which illustrates the country's diversity, despite its small size, not only in terms of languages (note: the distribution is showing numbers in our dataset, i.e. values per localities, hence the mean value in the boxplot is the mean of those not the mean of the individual votes in the canton).

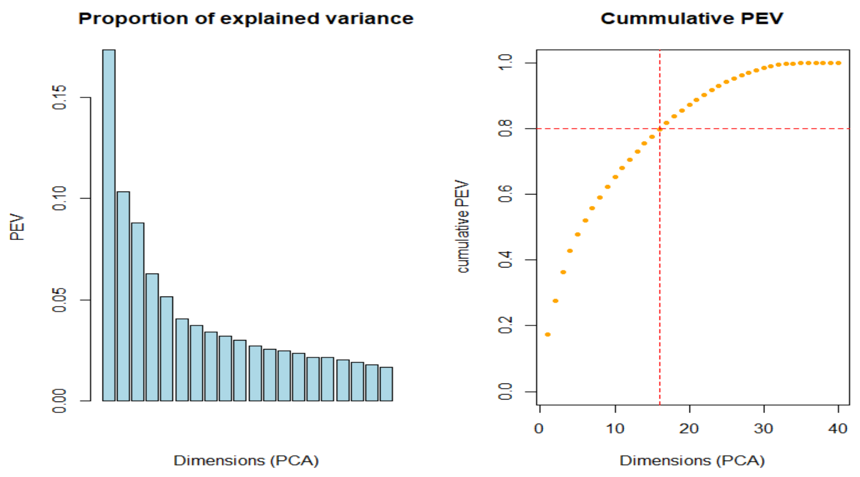


Another easy way to get insight of the results is to plot them by language. This already can hint at an answer to the "language" barrier question. Indeed, this indicates a significant difference between German speaking citizens' votes (D) and French ones (F), even more between French and Italian ones (I), surprisingly, as they both belong to the Latin language minorities. The Romansh case (R) is as well quite different from the rest, but marginal as it concerns very few localities (hence residents). We'll use later other techniques to confirm the importance of the language dimension in the voting results.

In order to get a deeper insight into our data, we will now be using two unsupervised learning methods: principal component analysis and K-means clustering. The first allows identifying what features contribute most to the overall variance in the data, building principal components that concentrate this variance, which allows reducing the number of features without losing much information. The second allows identifying clusters in the data that are closely located in the feature space, i.e. that present significant resemblance in terms of values.

Principal Component Analysis.

As we have a lot of variables to take into account (41 numerical features), we can consider our dataset as “wide” and it seems very sensible to perform a PCA analysis to check if the total variance can be explained by a smaller number of variables. Even if we excluded the very obvious correlated features, a few of the remaining ones look like they would present important correlation still, hence be good candidates for a dimension reduction.
To proceed with this, we exclude the target variables, as our goal here is to understand the importance of explanatory features. At this point, it is important to normalize the data (by setting the right parameter in the function call), because we want to identify what features contribute to the overall variance most, relatively to each other.



The images below illustrate the explained variance, as a function of the principal components. On the left, we can we that the first component only explains about 18% of variance, the second about 10% (we limited the display to the 20 most important values). On the right, we note that, in order to explain 80% of the variance, we need at least 16 components, displaying the cumulative proportion of explained variance.

We conclude that PCA enables to get rid of more than one half of the features and still explain 80% of the variance, which is interesting. However, the 16 remaining components are not particularly easy to interpret. We drew the components elements, i.e. linear combination of original dimensions: as they very numerous, it is hard to read and make sense of (the reader can however run the script to check the results).
As the PCA results will be helpful in the next analysis, we still try to make sense of the two main components, by checking how their composed, here are the top contributing features:

Highest contributors to component 1		Lowest contributors to component 1	
Primary.sector.emp	0.264	Settlement.and.urban.area.in.perc	-0.278
Agricultural.area.in.perc	0.121	Population.density.per.km.sq	-0.305
under.the.Swiss.Criminal.Code	0.118	Secondary.sector.establ	-0.325
Size.of.households.in.persons	0.112	Residents	-0.347

SVP	0.101	Tertiary.sector.establ	-0.348

Fig 3: Highest and lowest values in component 1

So the first component illustrates a clear dimension: large positive values show an agricultural locality, having residents working in the primary sector, in large households and belonging to the right-wing party (that happens to be the agrarian political party). Large negative values show, on the contrary, localities with numerous residents, densely populated, working in the tertiary sector, in an urban environment hosting many secondary sector establishments. Clearly, the first component contains the spectrum of urban to agricultural values, indicating population occupation sector.

Highest contributors to component 2		Lowest contributors to component 2	
Pop.65.years.or.over	0.354	Crude.birth.rate	-0.153
Unproductive.area.in.perc	0.322	GLP	-0.153
Total.surface.area.in.km.sq	0.282	Population.density.per.km.sq	-0.195
Crude.mortality.rate	0.220	Size.of.households.in.persons	-0.317
CVP	0.205	Pop.0.19.years	-0.341

Fig 5: Highest and lowest contributors to component 2

The second component seem to have large values when the population is aged, the locality is largely hosting unproductive land, is vast and has a large mortality rate. Negative values are due to high birth rate; people belong to the “green liberal” party, are young and belong to large households. So this component enable to place a given locality in a “socially dynamic” scale: the highest value indicating a population old and inactive and rather isolated, lowest value indicating localities densely populated, with a young population and many families.

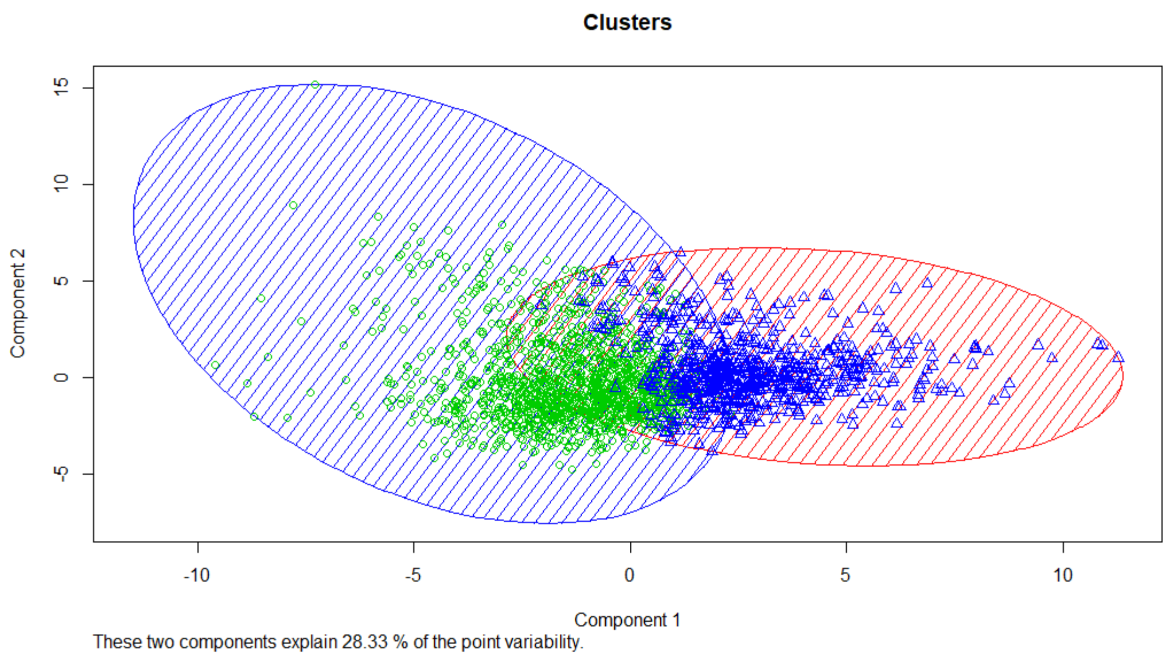
This makes sense, as Switzerland is indeed a very contrasted country, with densely populated areas, attracting young people, and more remote areas, typically in the mountains, where indeed land in unproductive and where people really attached to this more quiet and rural lifestyle tend to stay. In the first one, people tend to work in the tertiary and secondary sector, in the second one, in the primary sector.

As the volume of data treated here is rather small, hence we have no computational power issue, we believe there is little sense using PCA components for the rest of the analysis. On the contrary, this would blur the interpretation of further analysis, which will aim at understanding a clear relationship between the features and the target variable.

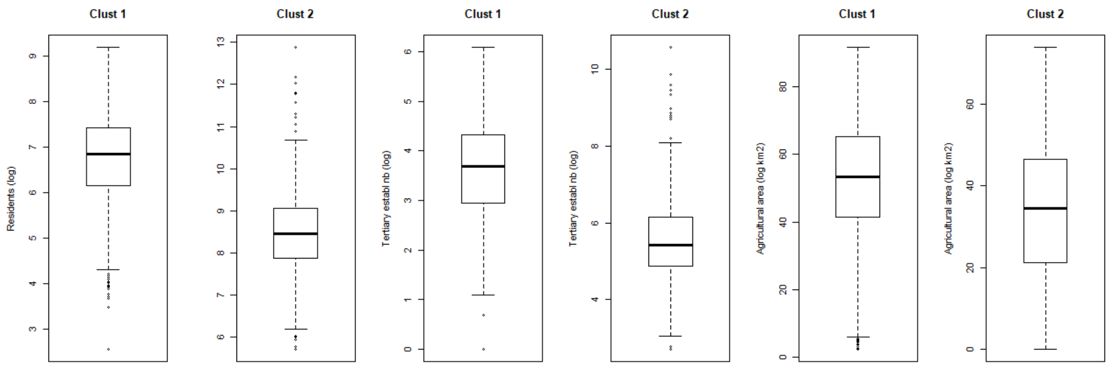
K-means Clustering.

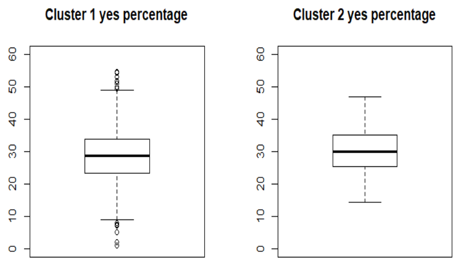
Clustering aims at identifying groups of objects that show similarities. It is an unsupervised machine learning method, as the target feature is not considered. In our case, we have eliminated it from the dataset (no voting data at all).
Therefore, all features are numeric here, which makes it straightforward to apply K-means clustering algorithm. K-means is center-based algorithm, well suited to large and high-dimensional data sets. Important is to normalize the input data, which is done by setting the function parameter in the call.

The key question when applying this algorithm is to check how many clusters there can be. Based on the PCA analysis, there seem to be at least one rather clear way to distinguish localities: the more urban ones versus the more rural ones. So we'll use 2 as number of cluster. The function returns the cluster memberships, centroids, sums of squares (within, between, total), and cluster sizes. Results are plotted below, along two axis that are the 2 PCA main components, explaining 28.33% percent of the variability. The results are two clusters that, according to the interpretation of those PCA made before, distinguish urban localities (in red area) with higher population density and people employed in tertiary sector, from rural ones (in blue area) whose population is working in the primary sector in a large proportion (the PCA dimension is actually inverted compared to the PCA above). One can note that the rural cluster has got more variability along component 2, measuring the "social dynamic" dimension previously explained. Interesting to note as well, the two clusters overlap, illustrating the fact that there are numerous "average" localities that are neither fully rural nor urban, illustrating a policy to keep an agricultural activity even in rather urban areas (Geneva's surrounding localities are a clear example of this).



At this stage, let's inspect come key dimensions in the first PCA to illustrate the actual difference in values distributions between the 2 identified clusters: number of residents (log), primary sector employees (%), and agricultural area (log of km2). The partition is clear for the 2 first ones, not so much for the last one, although the median is quite different. This confirms the clustering allows us to distinguish localities that we would call “urban” (red above) from “rural” ones (blue).





We can use this split to try and answer one of our questions: has the vote been significantly different within the two population subset: to do so, we display two boxplots showing the corresponding populations’ results percentages.
 Visually, no large difference is spotted, although a marginal one is showing the countryside localities (cluster 1) have accepted the proposal slightly less than the urban localities (cluster2). Maybe because isolated places inhabitants feel more the need to get in touch with the rest of society through TV and radio, this difference in the two groups does however not seem significant. We conclude that this assumption cannot be verified using our dataset: no clear difference is due to the urban/rural mentality “gap”.

4. Data analysis

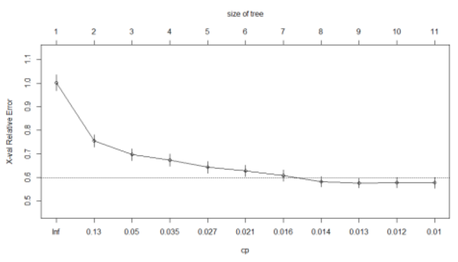
We are now addressing the regression problem: predict the voting result, i.e. percentage of “yes”, based on all other features values. One could wonder why we are trying to predict a result that we already have, of course. Mainly, building a successful regression model, provided it is comprehensible, will allow us to explain the results: why did such a locality yield such a result, based on the available data? If the prediction result is accurate, we’ll know we can trust the explanation provided by the model.
Another useful feature of such a model will be to spot where the model “fails”: for which localities does the prediction differ the most from the actual value? Sporting these elements will enable to dig deeper and look for anomalies in the data or other influencing factors, hinting at adding other exogenous data to the model, for example. This could also be very useful to identify fraud. Switzerland, like many other countries, is currently experimenting electronic vote. One of the major risks that is faced with this, is fraud by hacking the system, or exploiting other vulnerabilities, typically voter impersonation. Fraud could be detected by identifying prediction outliers: those would not be fraud cases for sure, but good candidates for further forensic analysis: there is a behaviour exception worth inspecting up close.

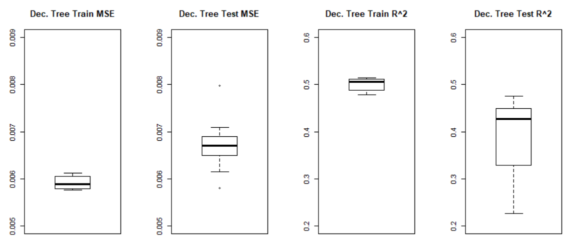
Regression using decision tree

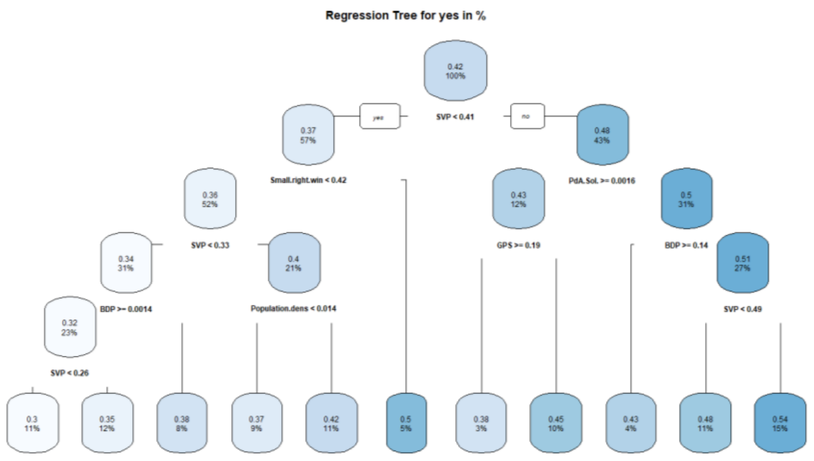
Using PCA and clustering analysis, we have seen that 2 clusters of localities could be identified, “urban” and “rural”, that had noticeable overlap. These clusters could not reveal a clear voting pattern however.

In order to identify such a pattern, based on the available information, we will perform an analysis using decision trees. Decisions trees provide results that are easy to understand: they recursively divide the input data into subsets where the response variable has small variance. In a regression problem, they predict an average value. Each split decision is taken by choosing an attribute and cut point that best reduces the regression errors. However, as the “growing” algorithm is greedy, there is no guarantee that is yields an optimal solution overall. On top they are subject to overfitting, which has to be remedied by performing appropriate pruning.

In order to run this algorithm, we consider the target feature, i.e. the voting result (the “yes” percentage). The rpart() package is used to create the tree, using the CART algorithm. It grows the tree using all the features present in the data. First, we encode numerically the language variable as the algorithm expects numerical values. The rpart() algorithm does the pruning automatically, by implementing a 10-fold cross validation. However, this does not suppress the need to split the data between train and test sets in order to verify the actual performance delivered using k-fold cross-validation as well (10 times, 90% split). The graph below shows the misclassification rate compared to the original score, per tree level, estimated from the cross validation (CART). We can see that a minimum is reached at 10 splits; therefore the pruning is done from there. On the right, we see that the average the R^2 value is 50% on the training set but 39% on the test set with considerable variance, which means that the model explains a less than half of the variability but tends to over fit and is not very robust (relatively high variances in R^2 results). This is not too bad, considering the nature of the problem; we therefore can confer some trust to this model. Let’s look at how it made its decisions.

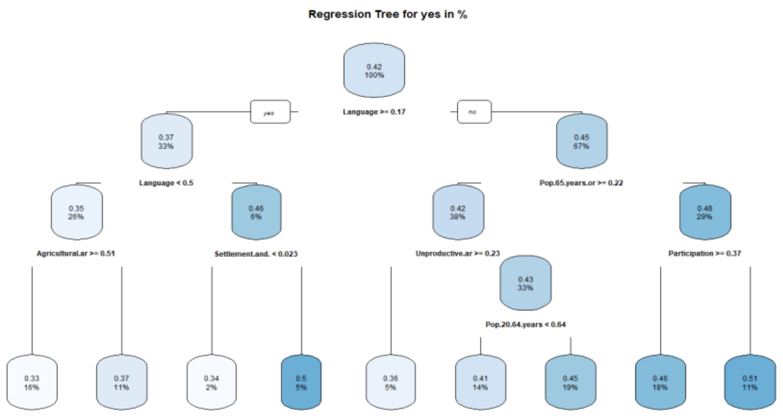






The resulting decision tree is very clear: the main criteria that can discriminate the result of the vote is the political affiliation, which makes a difference of 11% in the voting results! It is the criteria at first and second level, and throughout the tree, apart for one exception (population density). In a nutshell, the higher the percentage of residents belong the "SVP" (who advised to vote "yes) or other right-wing parties, the higher the vote towards the "yes". One could interpret this as "no-brainer": people tend to follow their party's advice and vote like they are told. Even if people did not really follow the advice, it makes sense that people who have affinities towards liberal parties would in principle rather support freedom of choosing what you pay for. On the other hand, the tree shows lowest percentages are clearly in localities where this far right party is least present. But this analysis doesn't provide any element about causality: we can't tell which come first, i.e. being part of a political party or having read the voting advice, or just having liberal affinities. All we can say is that there is a clear link in the data.

At this stage, we keep on analysing the data by removing the political information from the dataset, to check what factors are having a large impact after those. We train another tree and note that the predicting power has dropped by a large margin, at about 30%. Quite surprisingly, the results show that the language plays the most important role.



Language encoding: German =0, French =⅓, Italian =⅔, Romansh =1

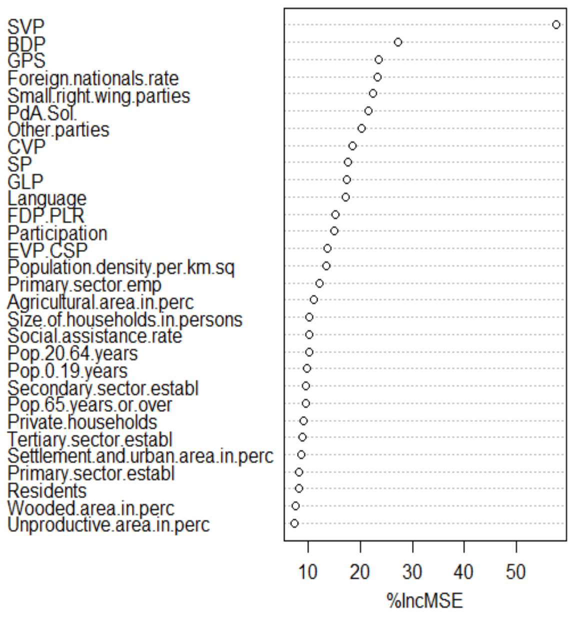
At this point, we can answer the "Rösti Graben" question (cultural gap due to language)! After political affiliation, it has clearly been a determining factor in this referendum: the German speaking has got a greater "yes" percentage than the other minorities (French/Italian/Romansh). Then, even more surprisingly, age plays a factor: the older generations (65 and older) have been accepting the abolition more than the younger ones: this contradict the idea that older people are attached to their TV channels as they tend to spend more time in front of it. This also means that the younger ones, who reputed to never watch TV but use Netflix and Youtube instead, did support their local traditional media. In this tree, although fewer features are used to build it, 9 splits are done before pruning is performed.

Regression using random forest

Random forests are believed to be one of the best "off-the-shelf" classifiers for high-dimensional data. It overcomes decision trees' main shortcoming, i.e. the "greedy" nature of their construction algorithm. In a regression tree model, terminal nodes reflect the mean of the observations in that node, obtained from the bagging process. Loss function used is MSE. We'll try to see how well it performs here, looking at its results for variables importance ranking and regression accuracy. Implementation is done with default number of trees (500).

Variables importance

A random forest allows us to determine the most important predictors across the explanatory variables by generating many decision trees and then ranking the variables by importance.



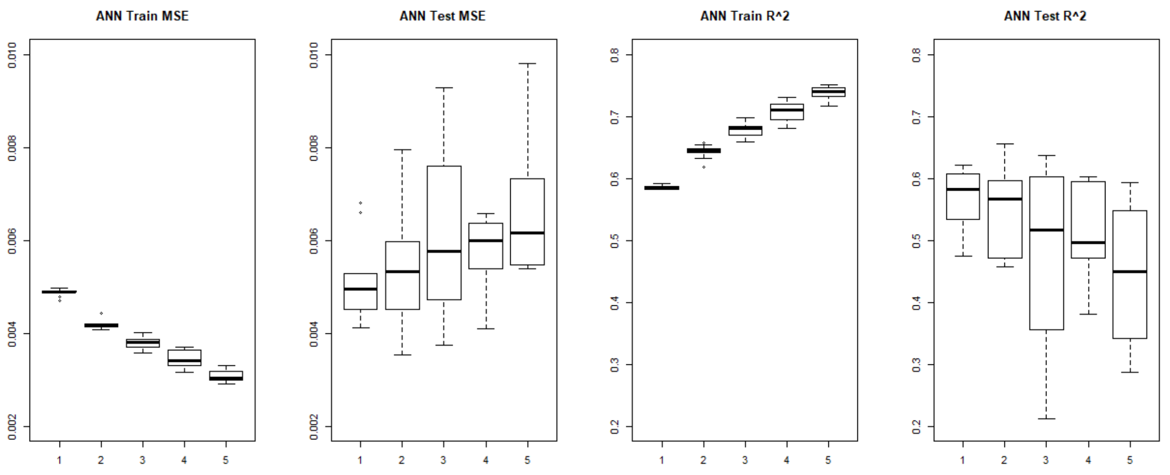
The graph on the left shows us features importance as delivered by the random forest algorithm. It is in line with what we found out in the decision tree exercise: the most important features are the political ones. Then it seems the foreign nationals rate and then the language. The greedy nature of the decision tree algorithm must have slightly overestimated the language importance. Note that this graph shows the importance of the features, it doesn't show how to interpret them, i.e. what decision they enable, like in the decision tree graphical representation.
As we will see below, the predictive power of the random forest is however remarkably higher than its decision tree counterpart, as expected.

Regression using artificial neural networks

As a final step to model the voting results from the explanatory variables, we will implement artificial neural networks. Their strength lies in their abilities to capture complex relationships between variables. However they are prone to overfitting. Like with decision trees, the goal will be to design the network that provides the best prediction accuracy without overfitting.

In order to get a representative accuracy for a given neural network model, we are implementing a cross-validation method, and splitting our dataset in train and test subsets, using a 90%/10% key. The cross-validation will redo the training/testing 10 times and deliver mean performance values (MSE and R^2). We will graph the variance of those results using boxplots to show related variability.

Designing a neural network model is often described as being more an art than a science. We'll proceed by testing several models, containing with 2 hidden layers: we are building a deep neural network. We'll automate the process of finding an optimal number of neurons for the first layer in terms of accuracy and overfitting.



The above graphs illustrate the ANN model performance, as a function of its number of neurons in the first hidden layer (the second contains 3). We can see that adding neurons improves performance on the training set, however this does not improve the test performance but does the opposite: there is a clear sign of overfitting as MSE grows with number of neurons. The optimal performance occurs with a first layer with only 1 neuron. It outperforms the decision tree by an important margin (MSE 31% better) and is close to random forest accuracy. We also note rather high variance in the test results, compared to the training results, indicating poor robustness. This is probably related to the relative small size of our dataset. A simpler topology without hidden layer was then tested, hinted by the one neuron layer performing best: it performs slightly worse. The simplest ANN with one neurone allows easy interpretability, since it has a unique set of weights that can be read to measure feature importance. If using ANN, we should prefer this simpler topology.

5. Performance evaluation

Performance designates here the “explaining power” of the model, or in other words how reliable it is in describing a phenomenon. As where are dealing with a regression problem, we need to make use of adapted performance measures: we pick MSE and R^2, which are the standard metrics for such problems and are standard for Random Forest regression (above). Mean Square Error measures overall error. The smaller it is the better. It will apply to our target feature. Note that is applies to scaled values at this point. The R squared measure, called coefficient of determination, is rather used in linear regression contexts; it is a measure of the explanatory power of the model. Between 0 and 1, it shows how much better the model is explaining variations than the simplest of models, i.e. the mean value (1 means perfect fit, it can actually be below 0 if model performs worse than the mean).

Method	**MSE**		**R squared**		**Train time [sec]**
	Train	**Test**	**Train**	**Test**	
Decision tree full	0.0059	0.0067	50.0%	39.0%	Not meas.
Dec. tree “apolitical”	0.0082	-	30,4%	-	Not meas.
Random forest	**0.0046**	-	**61.2%**	-	Not meas.

ANN (1,3)	0.0049	**0.0051**	58.6%	**56.6%**	2.46
ANN (2,3)	0.0042	0.0055	64.4%	55.1%	11.76
ANN (3,3)	0.0038	0.0063	67.8%	46.6%	22.98
ANN (4,3)	0.0034	0.0057	70.8%	51.6%	26.01
ANN (5,3)	0.0031	0.0066	73.8%	44.4%	14.95

Considering mainly the MSE, we see that the decision tree has got limited performance, and it is performing a lot worse when the most important features (i.e. the “political” ones) are removed. Random forest performs considerably better, as it produces the lowest MSE but its results can’t be easily explained; we did not check if was overfitting, as it should not by design. ANN can be made as complex as we want: the more neurons it has, the better it will calibrate using the input data, but it will also from its noise, and will not generalize well, as we clearly see above. It will also take more time to train. Hard to interpret however, the trend is not monotonous: MSE for 3 neurons is worse than for 2, but it improves again for 4. Same for execution time: it is shorter for 5 than for 4. We were convinced we could define an ANN topology that would “beat” the random forest: adding neurons or changing topology did never produce a model that could, to our surprise. To the extente that we cross checked our results with the Weka software to avoid any mistake in our code: results were consistent.

Outside accuracy, another key dimension to consider in our problem is results interpretability/ comprehensibility. We have seen that outside of the most readable model, i.e. decision tree, whose results are easy to understand for anyone, k-means clustering allows bringing clarity in the data, when graphing the results along the two main components as defined in the PCA analysis. As well, the random forest algorithm has allowed identifying very clearly which are the most important features, on top of its powerful predictive power. Finally, deep Artificial Neural Networks, even if they showed an interesting predictive power too, have lived up to their “black box” reputation and haven’t helped us clarity in our dataset or in any pattern it could contain, since their large quantity of weights and biases is too intricate to make sense of.

6. Discussion of the findings

Unsupervised learning findings

Using PCA, we have found out that the many features in our dataset could be reduced to only about one third and still explain 80% of the variance but at the cost of explicability.

Using clustering, we could show that indeed two “families” of localities could be distinguished, rural and urban, as one would expect. This could however not reveal any voting pattern, denying a city vs. countryside opposition on this matter.

Supervised learning findings

Using decision trees, we have been able to show what features can be used to estimate with a few simple tests the referendum results in a given locality. This illustrates the importance and meaning of the features in our dataset. However, this has a limited accuracy. We saw very clearly that political affiliation, taken globally, can determine with a certain degree of accuracy, the voting result of a given locality. This was not expected, as the media usually explain of the voting results by showing a “big cities” vs “countryside small towns” gap. We saw as well that language was playing a secondary role, probably because the French speaking minority had more to lose, giving credit to the “Rösti Graben” theory that is usually said to be a myth.

Using random forest, we have been able to measure features importance more accurately, producing a clear ranking and building a model with a high explanatory power, as it explains about 61% of the observed variation, using available features, and confirming the above observations. We have however not checked if it was overfitting.

Implementing artificial neural networks, we have tried to build a model able to capture complex relationships between features at best. Neural networks are often looked as a “black boxes”, they require more computation than other methods power: our project confirms this. We managed to design an ANN with the second smallest predictive error and that doesn’t over fit, when measured on test data, slightly worse than the random forest.

Conclusion

Switzerland is a small country full of contrasts and diversity. Its civic system has been designed to give its population direct access to political decisions, giving the word to everyone and respect minorities. Its diversity is well reflected in the referendum results, as we have seen.

As a conclusion, analysing the dataset we have been able to provide answers to our initial questions that rely on solid elements, delivered applying machine learning techniques on our dataset. We have been able to come to some surprising results, typically the weight of official political parties’ advice and debunked some myth: the “Rösti Graben” does exist, at least it did in this case and was more able to explain the vote than the urban/rural gap that is usually referred to in such cases.

Using all available features, we’ve even been able to build a model able to capture the voting behaviour; it does a rather good job since it does predict more than half of the variability (60%, random forest). We believe that we could use this predictive power to check in what cases the prediction is the worst, to focus analysis on those elements and spot eventual anomalies, in a fraud detection context.

On a philosophical note, our results show that we have not been able to find an “algorithm” that can predict vote results with a perfect accuracy, only using socioeconomics or political measures: although other techniques might outperform the ones implemented here, we believe it is unlikely the reach a much better accuracy. It is reassuring to think there still a part of mystery in human behaviour especially when it comes to taking decisions for the future of a country. Human still have some inner mystery that machines cannot read through, yet.

Appendix

R-code

```
######################################################################
#
#   CS5608 Big Data Analytics
#   Coursework "Swiss referendum results analysis"
#   Annotated code
#   Author: Christophe Cestonaro
#
######################################################################
#
#   0. Preparing environment
#   1. Loading and merging data
#   2. Exploratory analysis
#   3. Principal Component Analysis
#   4. Clustering analysis
#   5. Decision tree analysis
#   6. Random forest analysis
#   7. Neural networks analysis
#
######################################################################
# 0. Preparing environment
#

# Lot of features -> PCA shows it is hard to analyze all at once
#
# split analysis along 3 axes, that don't make sense if mixed up
# - language
# - socio-economics
# - political

# Define variable to execute script in a verbose manner, i.e. providing more info about tasks progress
# and detailed results. If set to FALSE, only most important info is printed
verboseExecution = TRUE

#
# Helper functions
#
#  function to check whether the package is required and install it if necessary
#  The function first builds the list of packages to be installed so it can inform users
pkg_install <- function(pkg_list){
  missingPackages <- vector()
  for( i in pkg_list ){
    #  checking 'require' to load package
    if(!require(i,character.only = TRUE)){
      missingPackages <- append(missingPackages, i)
    }
  }
  if (length(missingPackages)==0) {
    cat("No package needs to be installed\n")
  } else {
    cat("The following packages will be installed:")
    cat(missingPackages, sep=",")
    cat("\n")
    for( i in missingPackages ){
        install.packages(i,dependencies = TRUE)
        require(i,character.only = TRUE)
    }
  }
}

#  defining MinMax function used for resizing data intervals between 0 and 1
MinMax <- function(x){
  tx <- (x - min(x)) / (max(x) - min(x))
  return(tx)
}

# Prints the list of columns in the dataset passed as parameter along with their index
# Usefull to check the content of the dataset when it has many columns for debugging purposes
```

```
printCols <- function(dataset, verboseMode = FALSE) {
  if (verboseMode) {
    colnames <- colnames(dataset)
    for (i in (1:length(colnames))){
      cat(paste(i, colnames[i]),"\n")
    }
  }
}

#  call pkg_install with the list of package names
pkg_install(c("dplyr" , "e1071",  "cluster", "rpart", "rpart.plot","plyr",
              "neuralnet", "Metrics","stringr","ggcorrplot","corrplot", "FactoMineR","party", "arules",
"randomForest"))

# set working directory to project root. Needed to access data files
setwd("E:/__Brunel/__Big Data analytics/R scripts")

######################################################################
#1. Loading and wrangling data

# Load referendum results, per town. Conveniently, the first line in the file contains the columns' headers
# We set the stringsAsFactors argument to FALSE as we dont want strings to eb considered as categorical variables
RefResults <- read.csv("..\\Input data\\ResultsPerTownC.txt", header = TRUE, sep = "\t", fileEncoding="utf-16",
stringsAsFactors=FALSE)
# Load localities portrait. Header is included too. Contains socio-economics and political details per locality
CommunPortrait <- read.csv("..\\Input data\\CommunesPortraitC.txt", header = TRUE, sep = "\t", fileEncoding="utf-16",
stringsAsFactors=FALSE)
# Load language exceptions, i.e. localities that do not speak the same language as the rest of their cantom
CommunLanguageExcept <- read.csv("..\\Input data\\CommunesLanguageExceptions.txt", header = TRUE, sep = "\t",
stringsAsFactors=FALSE)

#--------------------------------------------------------- JOIN -----------------------------------------------------
#Join the two table: inner join is performed as lines with partial information is not needed (and is marginal)
ds <- inner_join(RefResults, CommunPortrait, by ='No.commune')
#Mergin with language exceptions:
ds <- left_join(ds,CommunLanguageExcept, by ='No.commune')

#Adding language information by cantons. D=German, F=French, I=Italian
ds[is.na(ds$Language),"Language"] <- "D" # As most communes speak German, we assign this value to all where missing
ds[ds$Canton=="Ticino","Language"] <- "I"
ds[ds$Canton=="Genève","Language"] <- "F"
ds[ds$Canton=="Neuchâtel","Language"] <- "F"
ds[ds$Canton=="Vaud","Language"] <- "F"

# Correct column names, to make them understandable, after R has replaced some caracters at import (like %)
colnames(ds)[colnames(ds)=="Yes.in.."] <- "Yes.in.perc"
colnames(ds)[colnames(ds)=="X.y"] <- "Commune.nb"
colnames(ds)[colnames(ds)=="Foreign.nationals.in.."] <- "Foreign.nationals.rate"
colnames(ds)[colnames(ds)=="Settlement.and.urban.area.in.."] <- "Settlement.and.urban.area.in.perc"
colnames(ds)[colnames(ds)=="Agricultural.area.in.."] <- "Agricultural.area.in.perc"
colnames(ds)[colnames(ds)=="Wooded.area.in.."] <- "Wooded.area.in.perc"
colnames(ds)[colnames(ds)=="Unproductive.area.in.."] <- "Unproductive.area.in.perc"

# Checking dimensions and basic statistics
if (verboseExecution) {
  #Referendum file contains 2228 towns (localities)
  dim(RefResults)     # Check number of rows and columns imported
  str(RefResults)     # Check that column types are what we expect. As it is fine, no need to force any to the
expected type.
  summary(RefResults) # Check values disribution and basic statistics

  #Towns "portrait" file contains 2324 towns (localities)
  dim(CommunPortrait) # Check number of rows and columns imported
  str(CommunPortrait) # Check that column types are what we expect.

  summary(CommunPortrait) # Check values disribution and basic statistics
  dim(ds)
  # common lines are 2200

  # we check for consistency by getting total number of voters, taht is also provided in original source excel file
  sum(ds["Total.voters"])
}

# Removing any columns that are duplicated (other common columns in the 2 input files)
```

```r
# As well as columns taht are useless in our analysis (changes in size size or numbers of citizen)
# and columns that are redundant, i.e. sum or % of values in other columns, as they don't
# bring any additional information
# 1 X.x
# 2 Commune
# 3 Total.voters
# 4 Total.ballots
# 5 Blank.ballots
# 6 Invalid.ballots
# 7 Valid.ballots
# 8 No
# 9 Commune.nb
# 10 Change.in..
# 11 Change.in.ha
# 12 Change.in.ha.1
# 13 Employed.total
# 14 Business.establishments.total (41)
# 15 New.housing.units.per.1000.residents (46)
# 16 Nom.Commune (62)

dropedCols <- c(1,5,6,7,9,10,11,13,15,18,32,34,37,41, 46, 62)
if (verboseExecution) {
  cat("Removing following columns:\n")
  printCols(ds[,dropedCols])
}
ds <- ds[,-dropedCols]

######################################################################
# 2. Exploratory Analysis

# Data Preparation
# inspect the dataset, check dimensions, values range and distribution
dim(ds)
str(ds)
summary(ds)

# Checking for duplicates
table(duplicated(ds)) # No duplicates found


if (verboseExecution) {
  # We check the top columns where the most important variance is observed
  list_std_dev <- apply(ds[8:47,],2,sd)
  head(sort(list_std_dev,decreasing = TRUE), 20)
}

# Basic statitics
# check 5 largest towns, as this dimension shows high variance and skewness
ds[(ds$Residents>100000),]
# total population
sum(ds$Residents) # 8097473
#average number of residents per locality
sum(ds$Residents)/length(ds) #172286.7

# generating a density plot for the votes to get an idea
plot(density(ds$Yes.in.perc), xlab="...")
# check the distribution by canton, to get a flavor
bp <- ggplot(ds, aes(x = Canton, y = Yes.in.perc)) +
  geom_boxplot(aes(colour = Canton), show.legend = FALSE)+
  theme(axis.text.x = element_text(angle =45))
bp
# at this stage, we can make faceted boxplots per language, to test if it plays a role
bp <- ggplot(ds, aes( y = Yes.in.perc)) +
  geom_boxplot(aes(colour = Language))+ facet_grid(. ~ as.factor(ds$Language))
bp

# As variance is very important in large towns (population or goegraphy), we fix this
# by applying a percentage of population measure instead of absolute number
ds$Primary.sector.emp <- ds$Primary.sector.emp/ds$Residents
ds$Secondary.sector.emp <- ds$Secondary.sector.emp/ds$Residents
ds$Tertiary.sector.emp <- ds$Tertiary.sector.emp/ds$Residents
ds$under.the.Foreign.Nationals.Act..FNA.<- ds$under.the.Foreign.Nationals.Act..FNA./ds$Residents
ds$under.the.Narcotics.Act..NarcA.<- ds$under.the.Narcotics.Act..NarcA./ds$Residents
ds$under.the.Swiss.Criminal.Code<- ds$under.the.Swiss.Criminal.Code/ds$Residents
```

```r
if (verboseExecution) {
  # check about skewness for number of residents dimension
  boxplot(log(ds$Residents))
  summary(log(ds$Residents)-2.5)
  # We check again in what columns the most important variance is observed
  list_std_dev <- apply(ds[8:47,],2,sd)
  head(sort(list_std_dev,decreasing = TRUE), 20)
}

CommunPortrait<- CommunPortrait[,-c(49)]

if (verboseExecution) {
  # check if log value present any skweness any more after log tranformation
  opar <- par(mfrow=c(2,6))
  # original distributions
  boxplot((CommunPortrait$Residents), main="Residents")
  boxplot((CommunPortrait$Private.households), main="Households")
  boxplot((CommunPortrait$Population.density.per.km.sq), main="Pop density")
  boxplot((CommunPortrait$Secondary.sector.establ+1), main="Secondary estab")
  boxplot((CommunPortrait$Tertiary.sector.establ), main="Tertiary estab")
  boxplot((CommunPortrait$Total.surface.area.in.km.sq), main= "area in km2")
  # distributions after log transformation
  boxplot(log(CommunPortrait$Residents), main="log")
  boxplot(log(CommunPortrait$Private.households), main="log")
  boxplot(log(CommunPortrait$Population.density.per.km.sq), main="log")
  boxplot(log(CommunPortrait$Secondary.sector.establ+1), main="log")
  boxplot(log(CommunPortrait$Tertiary.sector.establ), main="log")
  boxplot(log(CommunPortrait$Total.surface.area.in.km.sq), main="log")
  par(opar)
}

# Where a percentage cannot be applied, we use a log function to get a less skewed distribution
# (efficiency of this method has been checked visually above)
ds$Residents <- log(ds$Residents)
ds$Private.households <- log(ds$Private.households)
ds$Population.density.per.km.sq <- log(ds$Population.density.per.km.sq)
ds$Secondary.sector.establ <- log(ds$Secondary.sector.establ+1)
ds$Tertiary.sector.establ <- log(ds$Tertiary.sector.establ)
ds$Total.surface.area.in.km.sq <- log(ds$Total.surface.area.in.km.sq)

# Displaying correlations between numerical features
votCorr <- cor(ds[8:46],use="complete.obs")
opar <- par(mfrow=c(1,1))
corrplot(votCorr, method="color", type="upper", tl.col="black", tl.srt=45,  tl.cex = 0.5 )
par(opar)


######################################################################
# 3. Principal component analysis

# To build PCA dataset, dropping the non-numeric variables and the target features
dropedCols <- c(1,2,3,4,5,6,7)

if (verboseExecution) {
  cat("Removing following columns for PCA:\n")
  printCols(ds[,dropedCols])
}
train_set <- ds[,-dropedCols,]

# Turning the language values into numeric ones (1,2,3)
train_set$Language<-as.numeric(factor(train_set$Language))

# Perform dimension reduction (PCA)
# Scale features since they are measured in different scales, the PCA must be performed with standardized data (mean
= 0, variance = 1)
pc_ds <- prcomp(train_set, rank.=3, scale. = TRUE)
if (verboseExecution) {
  summary(pc_ds)
}
# calculate the proportion of explained variance (PEV) from the std values
pc_var <- pc_ds$sdev^2
pc_var
pc_PEV <- pc_var / sum(pc_var)
```

```
pc_PEV

pc_ds_loadings <- pc_ds$rotation
# To understand the 2 mains loadings, print out the 5 main elements, and 5 last ones
head(pc_ds_loadings[order(pc_ds_loadings[,1], decreasing =TRUE),1])
tail(pc_ds_loadings[order(pc_ds_loadings[,1], decreasing =TRUE),1])
head(pc_ds_loadings[order(pc_ds_loadings[,2], decreasing =TRUE),2])
tail(pc_ds_loadings[order(pc_ds_loadings[,2], decreasing =TRUE),2])

# plotting the explained proportion variance per PC, only 20 first values
opar <- par(mfrow=c(1,2))
barplot(
  pc_PEV[1:20],
  main="Proportion of explained variance",
  xlab="Dimensions (PCA)", ylab="PEV",col="lightblue")

plot(
  cumsum(pc_PEV),
  main = "Cummulative PEV",
  ylim = c(0,1),
  xlab = 'Dimensions (PCA)',
  ylab = 'cumulative PEV',
  pch = 20,
  col = 'orange'
)
abline(h = 0.8, col = 'red', lty = 'dashed')
abline(v = 16, col='red', lty = 'dashed') # 16 components needed to explain 80% of variance
par(opar)

#plotting the loadings
opar <- par(mfrow=c(1,1))
colvector = c('red', 'orange', 'yellow', 'green', 'cyan', 'blue', 'violet', 'brown')
labvector = c('PC1', 'PC2', 'PC3')
barplot(
  pc_ds_loadings[,c(1:3)],
  beside = T,
  yaxt = 'n',
  names.arg = labvector,
  col = colvector,
  ylim = c(-1,1),
  border = 'white',
  ylab = 'loadings'
)
axis(2, seq(-1,1,0.1))
legend(
  'bottomright',
  bty = 'n',
  col = colvector,
  pch = 15,
  row.names(pc_ds_loadings),
  cex=0.5
)
par(opar)

# Plot values along the 2 main components
opar = par(mfrow=c(1,1))
biplot(
  pc_ds,
  scale = 3,
  col = c('grey40','orange'), cex=0.5
)
par(opar)

######################################################################
#
#   4. Clustering analysis
#
######################################################################
# K-Means Clustering
# Net Income Adjustments and Earning per share fields are used to cluster
# data preparation: all data must be numerical and normalized
# noramlise the data
train_set_norm <- as.data.frame(apply(train_set, 2, MinMax))
```

```r
y <- train_set_norm[,-c(7)]

# distance matrix with euclidian distance
dist_ds <- dist(y, method = 'euclidian')
# k-means algorithm call. Assumption is that there are two clusters we can identify
k_ds <- kmeans(y,centers=2, nstart=20)
# inspect results
str(k_ds)
table(k_ds$cluster)

# inspect the 2 clusters, to try and make sense of them
cluster1 <-ds[k_ds$cluster==1,]
cluster2 <-ds[k_ds$cluster==2,]

# sort values by number of residents, to check if large towns are part of cluster 1
head(cluster2[order(cluster2$Residents, decreasing =TRUE),c("Name.of.commune","Residents",
"Population.density.per.km.sq", "Agricultural.area.in.perc", "Wooded.area.in.perc")],5)
tail(cluster2[order(cluster2$Residents, decreasing =TRUE),c("Name.of.commune","Residents",
"Population.density.per.km.sq", "Agricultural.area.in.perc", "Wooded.area.in.perc")],5)
# sort values by population density, to check if coutry-side localities are part of cluster 2
head(cluster1[order(cluster1$Population.density.per.km.sq, decreasing
=TRUE),c("Name.of.commune","Residents","Population.density.per.km.sq", "Agricultural.area.in.perc",
"Wooded.area.in.perc")],5)
tail(cluster1[order(cluster1$Population.density.per.km.sq, decreasing
=TRUE),c("Name.of.commune","Residents","Population.density.per.km.sq", "Agricultural.area.in.perc",
"Wooded.area.in.perc")],5)

# As clustering has split the locilities in cities vd country-side, check if this split explains referendum results
opar <-par(mfrow=c(1,2))
boxplot(cluster1$Yes.in.perc, ylim = c(0, 60), ylab ="Yes vote (%)", main="Cluster 1")
boxplot(cluster2$Yes.in.perc, ylim = c(0, 60), ylab ="Yes vote (%)", main="Cluster 2")
par(opar)

opar <-par(mfrow=c(1,6))
boxplot(cluster1$Residents, ylab ="Residents (log)",main="Clust 1")
boxplot(cluster2$Residents ,main="Clust 2")
boxplot(cluster1$Tertiary.sector.establ, ylab ="Tertiary establ nb (log)",main="Clust 1")
boxplot(cluster2$Tertiary.sector.establ, ylab ="Tertiary establ nb (log)", main="Clust 2")
boxplot(cluster1$Agricultural.area.in.perc, ylab ="Agricultural area (log km2)",main="Clust 1")
boxplot(cluster2$Agricultural.area.in.perc, ylab ="Agricultural area (log km2)",main="Clust 2")
par(opar)


# Plotting the k-means clusters
opar <- par(mfrow=c(1,1))
clusplot(y,
         k_ds$cluster,
         main = paste("Clusters"),
         labels=1,
         shade = TRUE,
         color = TRUE,
         col.p =k_ds$cluster+2,
         line = 0)
par(opar)

#######################################################################
#
#   5. Decision tree analysis
#
#######################################################################
# Point is to do a regression with decision tree, to check what factors are
# usefull to predict a referendu result and how

# dropping unneeded features, but not target value of course
dropedCols <- c(1,2,3,5,7)
if (verboseExecution) {
  cat("Removing following columns:\n")
  printCols(ds[,dropedCols])
}
train_set <- ds[,-dropedCols,]

# encode language feature numericaly and
train_set$Language<-as.numeric(factor(train_set$Language))
# normalize values, so large values dont get largest weight algorithm
```

```r
train_set_norm <- as.data.frame(apply(train_set, 2, MinMax))

# grow regression tree using rpart()
fit <- rpart(Yes.in.perc  ~ .,
             method="anova", data=train_set_norm)

printcp(fit) # display the results
opar <-  par(mfrow=c(1,1))
plotcp(fit) # visualize cross-validation results
par(opar)
summary(fit) # detailed summary of splits

# plot tree
opar <-  par(mfrow=c(1,1))
rpart.plot(fit,
     main="Regression Tree for yes in %",
     varlen = -15,
    cex = 0.75
     )
par(opar)

# calculate Rsquare
rsq.rpart(fit)
opar <- par(mfrow=c(1,2))
rsq.rpart(fit)
par(opar)

#Evaluate accuracy
res <- train_set_norm$Yes.in.perc - predict(fit,train_set_norm)
#MSE
mean(res^2) #0.006091819
# RMSE
sqrt(mean(res^2)) #0.07805011
# R-squared
1 - var(res) / var(train_set_norm$Yes.in.perc) #  0.4829105
#plot predicted values vs actual ones
plot(train_set_norm$Yes.in.perc,predict(fit,train_set_norm),col="blue",
     main = "Predicted values vs actual ones")
abline(c(0,1),col=2)

#plot predictions does not make much sense, as it is average values
plot(train_set_norm$Yes.in.perc,predict(fit,train_set_norm),col="blue")

#measure precision using k-fold validation, to check overfitting
errorTst <- vector()
RsqTst <- vector()
errorTrain <- vector()
RsqTrain <- vector()
train_set_norm_l <- train_set_norm%>%select(-2,everything())
for(i in 1:10){
  # split into test and train, using a 90% ratio (remember this is with the k-fold loop)
  index <- sample(1:nrow(train_set_norm_l),round(0.9*nrow(train_set_norm_l))) # Sample with replacement
  train <- train_set_norm_l[index,]
  test <- train_set_norm_l[-index,]
  # grow regression tree using rpart()
  fit <- rpart(Yes.in.perc  ~ .,
               method="anova", data=train)
  predictionsTest <- predict(fit, test[,-c(42),])   # compute predictions on test data
  predictionsTrain <- predict(fit, train[,-c(42),]) # compute predictions on train data
  residualTst <- test$Yes.in.perc - predictionsTest     # compute residuals on test data
  residualTrain <- train$Yes.in.perc - predictionsTrain # compute residuals on train data
  errorTst[i] <- mean(residualTst^2)                        # compute MSE on test data residuals
  RsqTst[i] <- 1 - (var(residualTst) / var(test$Yes.in.perc))      # compute R^2 on test data residuals and actual
values
  errorTrain[i] <- mean(residualTrain^2)                      # compute MSE on train data residuals
  RsqTrain[i] <- 1 - (var(residualTrain) / var(train$Yes.in.perc)) # compute R^2 on test train residuals and actual
values
}
# plot results
opar <-  par(mfrow=c(1,4))
boxplot(errorTrain, main = paste("Dec. Tree Train MSE"), ylim = c(0.005, 0.009))
boxplot(errorTst, main = paste("Dec. Tree Test MSE"), ylim = c(0.005, 0.009))
boxplot(RsqTrain, main = paste("Dec. Tree Train R^2"),ylim = c(0.2, 0.6))
boxplot(RsqTst, main = paste("Dec. Tree Test R^2"),ylim = c(0.2, 0.6))
```

```r
par(opar)

# Print out results of accuray metrics
cat("Decision tree accuracy results:\n")
cat("Train MSE:", mean(errorTrain[]),"\n")
cat("Test  MSE :", mean(errorTst[]),"\n" )
cat("Train R^2  :", mean(RsqTrain[]),"\n" )
cat("Test  R^2  :", mean(RsqTst[]),"\n" )

#same exercise but without political information, to check what other features are use next
train_set_norm_apolitical <- train_set_norm[,-c(28:38)]
str(train_set_norm_apolitical)
# grow tree
fit <- rpart(Yes.in.perc  ~ .,
             method="anova", data=train_set_norm_apolitical)

printcp(fit) # display the results
plotcp(fit) # visualize cross-validation results
summary(fit) # detailed summary of splits

par(mfrow=c(1,2))
rsq.rpart(fit)

# plot tree
opar <-  par(mfrow=c(1,1))
rpart.plot(fit,
           main="Regression Tree for yes in %",
           varlen = -15,
           cex = 0.75
)
par(opar)

#Evaluate accuracy
res <- train_set_norm$Yes.in.perc - predict(fit,train_set_norm_apolitical)
#MSE
mean(res^2) #0.008200373
# RMSE
sqrt(mean(res^2)) #0.07805011
# R-squared
1 - var(res) / var(train_set_norm$Yes.in.perc) #  0.303931

#plot predicted values vs actual ones (ground truth)
plot(train_set_norm$Yes.in.perc,predict(fit,train_set_norm_apolitical),col="blue",
     main = "Predicted values vs actual ones")
abline(c(0,1),col=2)

######################################################################
#
#   6. Random forest analysis
#
######################################################################

# we now use random forest to check features importance
train <- sample(1:nrow(train_set_norm),2000)
set.seed(101) # for reproducibility
rf <- randomForest(Yes.in.perc ~ . , data = train_set_norm , subset = train, importance=TRUE)
print(rf)

opar <-  par(mfrow=c(1,1))
plot(rf, main = "Random forest error")
par(opar)

#check where the error is minimal, in terms of
which.min(rf$mse) #-> 489

#Evaluate variable importance
importance(rf)
varImpPlot(rf, main = "Feature importance in random forest")

plot(train_set_norm$Yes.in.perc,predict(rf,train_set_norm),col="blue",
     main = "Predicted values vs actual ones")
abline(c(0,1),col=2)

######################################################################
```

```
#
#   7. Neural networks analysis
#
######################################################################

# Neural network
# to be consistent with usual practice, we move the target feature to the last column
train_set_norm <- train_set_norm%>%select(-2,everything())

# Implementing cross validation in order to measure accury as well as possible
set.seed(2019)
nn.ExecTime <- NULL
k <- 10 # Using 10 folds cross validation
m <- 1 # initial numbeer of neurons in second layer
l <- 5 # Max number of neurons in second layer$

nn.errorTst <- matrix(, nrow = k, ncol = l-m+1)
nn.R2Tst <- matrix(, nrow = k, ncol = l-m+1)
nn.errorTrain <- matrix(, nrow = k, ncol = l-m+1)
nn.R2Train <- matrix(, nrow = k, ncol = l-m+1)

pbar <- create_progress_bar('text') # Implementing progress bar as process takes time
pbar$init(k*(l-m+1))
for (j in m:l){
  cat("Training network with ",j,",3 hidden neurons.\n")
  for(i in 1:k){
    # Keep track of time spent
    start.time <- Sys.time()
    # split into test and train, using a 90% ratio (remember this is with the k-fold loop)
    index <- sample(1:nrow(train_set_norm),round(0.9*nrow(train_set_norm))) # Sample with replacement
    train.cv <- train_set_norm[index,]
    test.cv <- train_set_norm[-index,]
    # Perceptron has 2 layers with 3 in the second. First layer has from m to l
    nn <- neuralnet(Yes.in.perc ~ . ,data=train.cv,hidden=c(j,3))
    predictionsTest  <- compute(nn, test.cv[,-c(42),])   # compute predictions on test data
    predictionsTrain <- compute(nn, train.cv[,-c(42),])  # compute predictions on train data
    residualTst   <- test.cv$Yes.in.perc  - predictionsTest$net.result     # compute residuals on test data
    residualTrain <- train.cv$Yes.in.perc - predictionsTrain$net.result    # compute residuals on train data
    nn.errorTst[i,j-m+1]   <- mean(residualTst^2)                          # compute MSE on test data residuals
    nn.errorTrain[i,j-m+1] <- mean(residualTrain^2)                        # compute MSE on train data residuals
    nn.R2Tst[i,j-m+1]      <- 1- var(residualTst)   / var(test.cv$Yes.in.perc)  # compute R^2 on test data residuals
and actual values
    nn.R2Train[i,j-m+1]    <- 1- var(residualTrain) / var(train.cv$Yes.in.perc) # compute R^2 on train data residuals
and actual values
    pbar$step()             # Advance progress bar
    end.time <- Sys.time()  # Compute elapsed time
    nn.ExecTime[i] <- end.time - start.time
  }
  # Print out results of accuray metrics
  cat("Results for layer:",j,"\n")
  cat("Train MSE  :", mean(nn.errorTrain[,j-m+1]),"\n")
  cat("Test  MSE  :", mean(nn.errorTst[,j-m+1]),"\n" )
  cat("Train RMSE :", sqrt(mean(nn.errorTrain[,j-m+1])),"\n" )
  cat("Test  RMSE :", sqrt(mean(nn.errorTst[,j-m+1])),"\n" )
  cat("Train R^2  :", mean(nn.R2Train[,j-m+1]),"\n" )
  cat("Test  R^2  :", mean(nn.R2Tst[,j-m+1]),"\n" )
  cat("Time spent :", mean(nn.ExecTime), "\n" )
}

# plot results, i.e. accuracy measures for a ANN, as boxplots
opar <-  par(mfrow=c(1,4))
boxplot(nn.errorTrain, main = paste("ANN Train MSE"), names = c(1,2,3,4,5), ylim = c(0.002, 0.01))
boxplot(nn.errorTst, main = paste("ANN Test MSE"), names = c(1,2,3,4,5), ylim = c(0.002, 0.01))
boxplot(nn.R2Train, main = paste("ANN Train R^2"), names = c(1,2,3,4,5),ylim = c(0.2, 0.8))
boxplot(nn.R2Tst, main = paste("ANN Test R^2"), names = c(1,2,3,4,5),ylim = c(0.2, 0.8))
par(opar)

# Environment dump, for reproducibility
#
# sessionInfo()
# R version 3.5.1 (2018-07-02)
# Platform: x86_64-w64-mingw32/x64 (64-bit)
# Running under: Windows >= 8 x64 (build 9200)
#
```

```
# Matrix products: default
#
# locale:
#   [1] LC_COLLATE=French_Switzerland.1252  LC_CTYPE=French_Switzerland.1252    LC_MONETARY=French_Switzerland.1252
# [4] LC_NUMERIC=C                        LC_TIME=French_Switzerland.1252
#
# attached base packages:
#   [1] stats4    grid      stats     graphics  grDevices utils     datasets  methods   base
#
# other attached packages:
#   [1] rpart.plot_3.0.7    arules_1.6-3        Matrix_1.2-14       party_1.3-3         strucchange_1.5-1
# [6] sandwich_2.5-0      zoo_1.8-4           modeltools_0.2-22   mvtnorm_1.0-10      FactoMineR_1.41
# [11] corrplot_0.84       ggcorrplot_0.1.2    ggplot2_3.1.0       stringr_1.3.1       Metrics_0.1.4
# [16] neuralnet_1.44.2    rpart_4.1-13        cluster_2.0.7-1     e1071_1.7-1         dplyr_0.8.0.1
# [21] randomForest_4.6-14 tree_1.0-39
#
# loaded via a namespace (and not attached):
#   [1] tidyselect_0.2.5     coin_1.3-0           reshape2_1.4.3       purrr_0.3.1          splines_3.5.1
# [6] lattice_0.20-35      colorspace_1.3-2     survival_2.42-3      rlang_0.3.1          pillar_1.3.1
# [11] glue_1.3.0           withr_2.1.2          matrixStats_0.54.0   multcomp_1.4-10      plyr_1.8.4
# [16] munsell_0.5.0        gtable_0.2.0         leaps_3.0            codetools_0.2-15     labeling_0.3
# [21] parallel_3.5.1       class_7.3-14         TH.data_1.0-10       Rcpp_1.0.0           scales_1.0.0
# [26] flashClust_1.01-2    scatterplot3d_0.3-41 digest_0.6.18        stringi_1.2.4        tools_3.5.1
# [31] magrittr_1.5         lazyeval_0.2.1       tibble_2.0.1         crayon_1.3.4         pkgconfig_2.0.2
# [36] MASS_7.3-50          libcoin_1.0-4        assertthat_0.2.0     R6_2.3.0             compiler_3.5.1
```