Abstract

In Republic of Lithuania, public elects their representatives to a parliament in which new legislation is considered. Due nature of politics, a problem arises once citizens wants to observe and evaluate parliament members work. Main output of parliament is their votes for various legislatures. Voting data is publicly available through Parliament of the Republic of Lithuania (LRS) website. This project aims to visualize voting patterns and their changes. Another goal is to provide public access to results.

In the project Multidimensional scaling (MDS) and *k-means* clustering are used to analyze voting patterns. Software to download, process and visualize data is written with Scala.

Santrauka

Atstovus į Lietuvos Respublikos Seimą (LRS), kuriame yra apsvarstomi ir priimami nauji įstatymai, išrenka Lietuvos piliečiai. Tačiau iškyla problemų visuomenei norint iš arčiau stebėti ir įvertinti Seimo narių darbą. Viena pagrindinių Seimo narių pareigų ir yra balsavimas priimant įvairius naujus įstatymus ar jų pataisymus. Kiekvieno Seimo nario balsai yra viešai prieinami Lietuvos Respublikos Seimo internetinėje svetainėje. Šiuo darbu siekiama šią informaciją vizualizuoti, kad būtų galima stebėti balsavimo modelius ir jų pasikeitimus. Taip pat yra siekiama suteikti viešą prieigą prie šių rezultatų.

Darbe, analizuojant balsavimo modelius, yra naudojamas daugiadimencinis skalių metodas ir k-reikšmių klasterizacija. Programinė įranga, duomenų atsisiuntimas ir vizualizavimas yra parašytas naudojantis Scala.

Contents

Acronyms

API Application programming interface. 19

LRS Parliament of the Republic of Lithuania. i, 2, 3, 19

MDS Multidimensional scaling. i, 1, 2, 9–12, 19, 21, 23, 25–27, 29–31

SQL Structured Query Language. 18

XML PLACEHOLDER. 3

Glossary

k-means . i, 1, 9, 10, 12, 19, 25–31

atviras-seimas.lt . 11, 12, 14, 15

dissimilarity matrix . 2

draw.io . 13

Euclidian distance . 19

HTTP . 18

JVM . 18

LRS open data Open data of Lietuvos Respublikos Seimas. 11, 19

Mockup description. 13

MySQL is an open-source relational database management system . 3, 18

Scala Scala is a general-purpose programming language providing support for functional programming and a strong static type system . i, 18

1 Introduction

In Republic of Lithuania, public elects their representatives to a parliament in which new legislation is considered. By this election each citizen delegates specifics of legislation process to their representatives so they don't have to actively participate in the process.

However, a problem arises once citizen wants to validate what his representative has been doing. Single term of office NEEDS-CITATION involves thousands of complicated laws and votes. To analyze everything becomes almost an impossible task for a single citizen who is out of the loop.

To make it easier there are journalists, politologists and other personas who review new legislation, current issues. Delegates themselves also do press conferences, debates where they state their intentions, comment on their actions. However, this requires citizens to trust that journalists and delegates only state truth, don't omit important information and don't have other hidden agendas. Study done about intrinsic honesty showed that the more society is corrupt - the more people lie in a simple dice game. This applies to politicians too and citizens trust in parliament is relatively low. Therefore, anything that can be done to better observe representatives is useful.

This project aims to deliver a software system which can be accessed online with research done available to any user. Research will include visualizations of Multidimensional scaling (MDS) for specific time periods of a given term. *K-means* projected on MDS output to visualize minority vs majority and to see if there are secret groups.

2 Analysis

2.1 Literature analysis

There is a decent amount of previous work analyzing voting on roll call data. A huge part of this research is on specific elections that happened in the past.

In *Spatial Models of Parliamentary Voting* [1] author discusses how voter's positions on specific issues can be captured by his position on one or two dimensions such as liberalism or conservatism. This constraint means there are two spaces - one with few dimensions - basic or ideological. The other - high dimensional space which represents remaining issues. This breakthrough might suggest Multidimensional scaling as a good performance method for visualization and analysis as majority of data is encoded in few dimensions.

There is research done specifically on LRS data. One such is *On Structural Analysis of Parliamentarian Voting Data* [2]. In this paper authors discuss about data reduction to dissimilarity matrix, vote encoding, MDS and its performance on specific dimensions. Authors focus on specific elections and term of office which is different from our goal. However, methods discussed and research results are relevant for this project.

Master thesis was written few years ago on LRS voting which included similar analysis. One of research areas was to see if there are hidden groups in the parliament [3]. Author was not able to detect hidden groups. In this project, goal, data and parameters are different so conclusion might differ too. Method used was *k-means* clustering and data was showed on MDS reduced coordinates. In this project similar approach can be taken with different parameters like date ranges for votes.

In *The new Voteview.com: preserving and continuing Keith Poole's infrastructure for scholars, students and observers of Congress* [4] paper, authors discuss famous *Voteview.com* website. While website's primary goal is to provide open data access is different from ours - it contains useful information about how specific methods are used, how visualizations work. It also contains visualization which shows how data changed over time - how ideology and party composition changes.

2.2 Materials and methods

2.2.1 Lithuania's parliament open data semantics analysis

In this section open data from XML is reviewed. Only data and properties important to the project are included and commented on. All data is imported into MySQL database, therefore to demonstrate structure tables are used. Field types are inferred empirically by looking at data, therefore might not be correct in some cases. Moreover, data itself is not consistent through all years of Parliament of the Republic of Lithuania activity.

Table 1 contains term of office table structure.

In XML	In database	Type	Comments
kadencijos_id	term_of_office_id	int(11)	Unique term of office id
pavadinimas	name	varchar(255) not null	Name
data_nuo	date_from	date not null	Date when term of office begins
data_iki	date_to	default null	Date when term of office ends

Table 1: Term of office table structure

Table 2 contains parliament sessions table structure.

In XML	In database	Type	Comments
sesijos_id	session_id	int(11) not null	Unique parliament session id
kadencijos_id	term_of_office_id	int(11)	Unique term of office id
numeris	number	varchar(255) not null	Number
pavadinimas	name	varchar(255) not null	Session name
data_nuo	date_from	date not null	Date when session begins
data_iki	date_to	date default null	Date when session ends. Current session doesn't have this value set.

Table 2: Parliament session table structure

Table 3 contains parliament plenaries table structure.

In XML	In database	Type	Comments
posėdžio_id	plenary_id	int(11) not null	Unique plenary id
sesijos_id	session_id	int(11) not null	Unique parliament session id
numeris	number	varchar(255) not null	Number
tipas	plenary_type	varchar(255) not null	Plenary type
pradžia	time_start	datetime default null	Plenary starting time
pabaiga	time_finish	datetime default null	Plenary ending time. Some plenaries are not yet finished and some entries don't include times at all

Table 3: Plenary table structure

Table 4 contains parliament member table structure.

In XML	In database	Type	Comments
asmens_id	person_id	int(11) not null	Person id
-	unique_id	varchar(100) not null	Unique id for person, generated by script
vardas	person_name	varchar(255) not null	Member name
pavardė	person_surname	varchar(255) not null	Member surname
lytis	gender	varchar(1) not null	Gender
data_nuo	date_from	date not null	Date from inauguration
data_iki	date_to	date default null	Date when term ends. Current session doesn't have this value set.
iškėlusi_partija	faction_name	text	Faction name
išrinkimo_būdas	elected_how	varchar(255) not null	How member was elected
biografi-jos_nuoroda	biography_link	varchar(255) default null	Hyperlink to biography
-	term_of_office_id	int(11)	Unique term of office id
-	term_of_office_specific_id	int(11)	Specific term id used for computations
-	faction_id	int(11) not null	Faction id

Table 4: Parliament member table structure

Table 5 contains agenda question table structure.

In XML	In database	Type	Comments
darbotvarkės_klausimo_id	agenda_question_id	int(11) not null	Agenda question id
-	agenda_question_group_id	varchar(255) not null	Group id
pavadinimas	title	text	Title
laikas_nuo	time_from	time default null	Question start time
laikas_iki	time_to	time default null	Question end time
-	datetime_from	datetime default null	Custom generated date time
-	datetime_to	datetime default null	Custom generated date time
data	date	date not null	Date
pavadinimas	raw_status	varchar(255) not null	Status as taken from XML
-	status	int(5) default null	Status encoded
dokumento_nuoroda	document_link	varchar(255) default null	Hyperlink to document
-	speakers	text	Speakers list converted from XML to single text field with separators
numeris	number	varchar(255) not null	Number
-	plenary_id	int(11) not null	Plenary id

Table 5: Agenda question table structure

Table 6 contains plenary question table structure.

In XML	In database	Type	Comments
darbotvarkės_klausimo_id	agenda_question_id	int(11) not null	Agenda question id
-	plenary_question_group_id	varchar(255) not null	Group id
pavadinimas	title	text	Title
laikas_nuo	time_from	time default null	Question start time
-	datetime_from	datetime default null	Custom generated date time
pavadinimas	raw_status	varchar(255) not null	Status as taken from XML
-	status	int(5) default null	Status encoded
numeris	number	varchar(255) not null	Number
-	plenary_id	int(11) not null	Plenary id

Table 6: Plenary question table structure

Table 7 contains discussion event table structure.

In XML	In database	Type	Comments
-	agenda_question_id	int(11) not null	Agenda question id
-	unique_id	varchar(255) not null	Custom generated unique id
laikas_nuo	discusstion_time_from	time default null	Event start time
asmens_id	person_id	nt(11) not null	Person id
balsavimo_id	vote_id	int(11) not null	Vote id
balsavimo_tipas	vote_type	int(5) default null	Vote type
-	plenary_id	int(11) not null	Plenary id

Table 7: Discussion event table structure

Table 8 contains vote table structure.

In XML	In database	Type	Comments
-	vote_person_id	int(11) not null	Custom generated unique id for vote
balsavimo_laikas	time	time DEFAULT not null	Voting time
balsavo	vote_total	int(11) not null	How many voted
viso	vote_total_max	int(11) not null	How many can vote
už	vote_for	int(11) not null	Voted for
prieš	vote_against	int(11) not null	Voted against
susilaikė	vote_abstain	int(11) not null	Voted abstain
komentaras	comment	text	Vote comment
asmens_id	person_id	int(11) not null	Person id
asmeyns_vardas	person_name	varchar(255) not null	Person name
asmens_pavarde	person_surname	varchar(255) not null	Person surname
kaip_balsavo	vote	int(11) not null	Vote itself
-	vote_person_id	int(11) not null	Custom generated unique id for vote
frakcija	faction_acronym	varchar(255) default null	Acronym for faction
balsavimo_id	vote_id	int(11) not null	Vote id
balsavimo_tipas	vote_type	int(5) default null	Vote type
-	vote_id	int(11) not null	Vote id
-	plenary_id	int(11) not null	Plenary id

Table 8: Votes table structure

2.3 Method analysis

2.3.1 MDS

Multidimensional scaling (MDS) is a method to visualize similarity of individual cases for a specific dataset. It takes an input matrix called proximity matrix Υ (or distance matrix). Proximity matrix is symmetric matrix containing distances between all objects ψ. Or in more generic terms - it contains similarities or dissimilarities of pairs. Actual distance can be calculated using Euclidian distance as described in formula 1. Classic MDS will assume distance to be Euclidian.

$$d(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{1}$$

Method outputs coordinate matrix minimizes loss function *strain* [5]. Given proximity matrix Υ, goal of MDS is to find ψ vectors $q_i,...,q_\psi$ such that $||q_i - q_j|| \approx \upsilon_{ij} \forall i,j \in 1,...,\upsilon$, where $||x||$ is vector norm.

2.3.2 Unsupervised learning: *k*-means clusterization

k-means performance can be measured two ways: manually and with *distortion*. It is possible to inspect clusters and see if results given are somewhat correct. *Distortion* is final sum of the squared distances between each vector and its centroid [6].

2.4 Problem analysis

Problem can be divided into two pieces:

- Research part, including MDS and *k-means* classification
- Software development

If considered what is output of parliaments during their term of office in parliament - it would be their voting outcomes. Let's say there is a set of all votes made by parliament members $V = \{v_1, v_2, ...v_n\}$ where each vote v_i has a tuple of parameters $P = \{timestamp, term\ of\ office, parliament\ member\ id, voting\ outcome, ...\}$. Votes can be divided into groups depended on term of office T during which they were cast. Each term of office T_i can be divided into smaller arbitrarily chosen time periods $\{p_1, p_2, ...p_n\}$. Each time periods can have votes assigned which were cast during that time. Goal is visualize set V in a way that similar voting patterns of different members are visible.

In order to visualize voting patterns by members, its dimensions need to be reduced. With help of Multidimensional scaling (MDS), vote set V with parameters set P can be visualized on a two or one dimensional coordinate system. As discussed in literature analysis and [1] voting patterns can be captured by only few dimensions. To observe how voting patterns change during time, different time periods could be chosen. Term of office should be the maximum time period p_n to analyze, as majority and minority usually changes together with parliament members.

Another goal is to see if votes can be classified to certain political groups' without knowing who voted. This also enables us to see if there are hidden factions between the explicitly stated ones. As discussed in literature analysis, in order to predict if voting pattern reveals political faction, majority or minority side, different amount of clusters need to be chosen. Finding clusters $\{c_1, c_2, ...c_n\}$ representing political faction voting patterns and then assigning votes to the specifics clusters will show how similarly explicitly stated sides vote.

3 Design and development of atviras-seimas.lt

3.1 Components of system

3.1.1 Functional requirements

Main goal of software is to provide public access for masses to view analysis and research done in this project. Software will run on virtual server and be available online on atviras-seimas.lt. There are two main roles: guest user and scientist. Users will be able to access system with any modern browser. Scientist, or the person maintaining project will be able to update data to most recent one on LRS open data website.

Data is visualized using interactive diagrams. In table 9 functional requirements are presented.

Requirement	User	Scientist
Access online with browser	✓	✓
View MDS results	✓	✓
Filter MDS results	✓	✓
Select different time periods for MDS results	✓	✓
View clustering results	✓	✓
Filter clustering results	✓	✓
Automatically download data with http query		✓
Automatically compute data with http query		✓

Table 9: Functional requirements for atviras-seimas.lt

3.1.2 Non-functional requirements

Non functional requirements:

- Page should load faster than 2 seconds
- Data should update in less than a day after query running
- User experience for users should be good
- System source code should be open source

3.1.3 Data flow diagram

On atviras-seimas.lt software system data flow is visualized in figure 1. Scientist can initiate coordinator to start *Downloader* which will download data and clean data depending on the scientist query. Scientist can also initiate query which will process data with MDS or *k-means*.

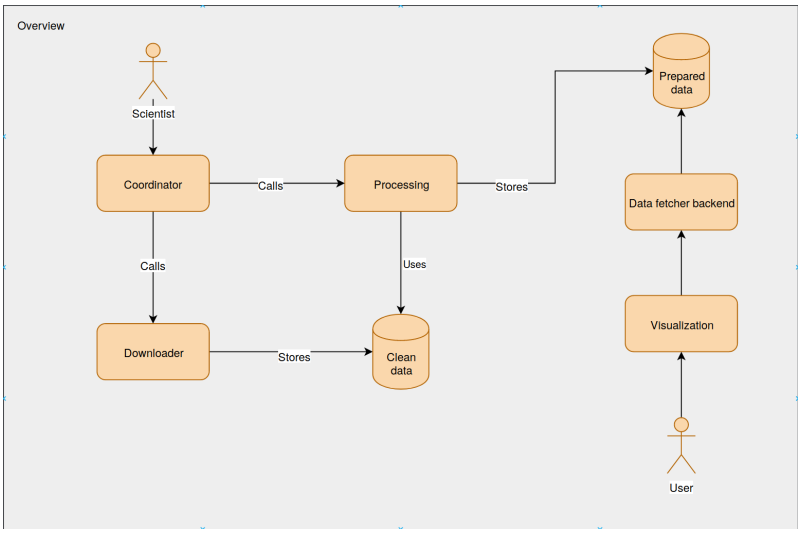


Figure 1: Data flow from database to *k-means*

3.2 User interface

3.2.1 User flow

Mockups are created for desktop size application with draw.io diagramming tool.

Happy path NEEDS CITATION is x and y, copy from scatter plot.

. It can and be viewed in figures 2 and 3. In figure 2 it is visible that when user clicks landing page button, user is redirected to second screen with two tabs. Tab content can be visible together with filter controls. Clicking on tabs results into switching to another tab and can be seen on figure 3.

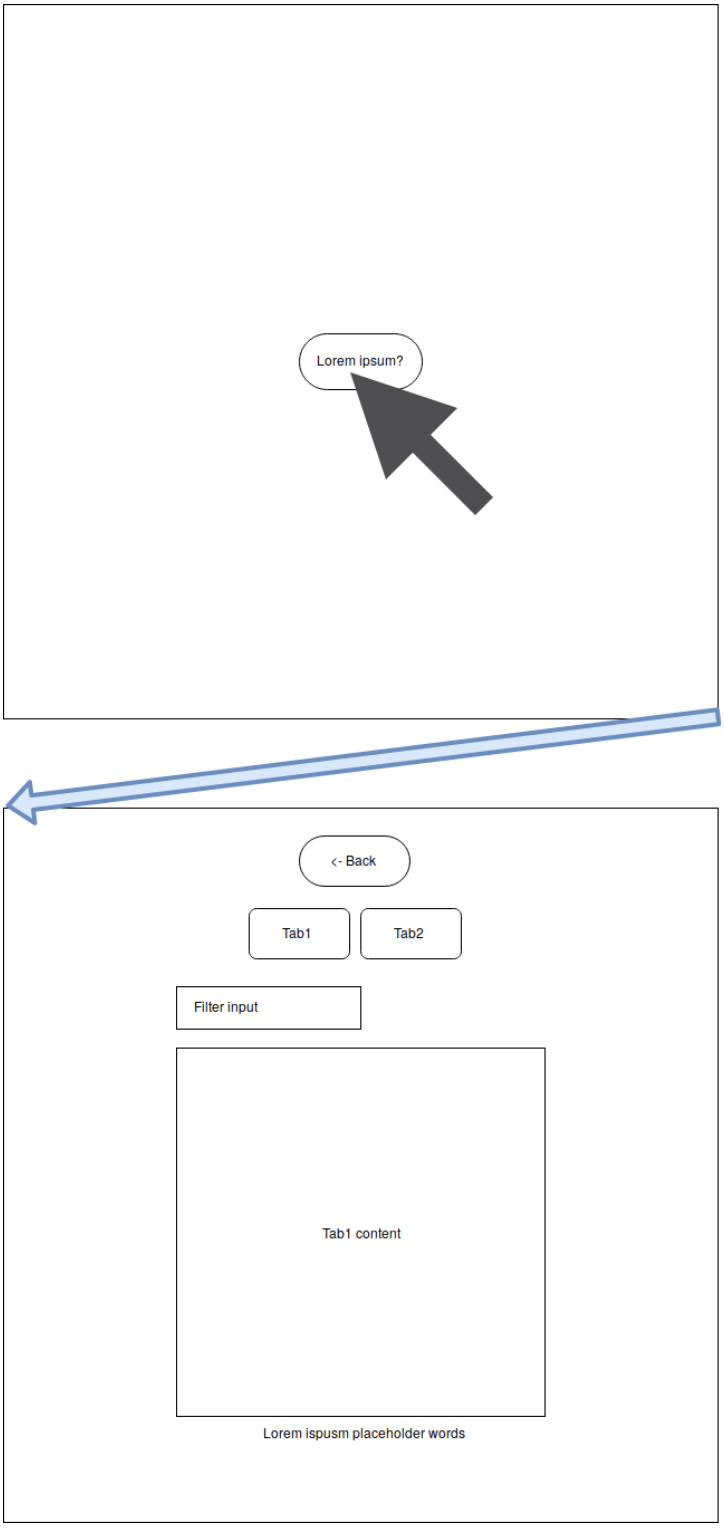


Figure 2: Mockup of atviras-seimas.lt first screen

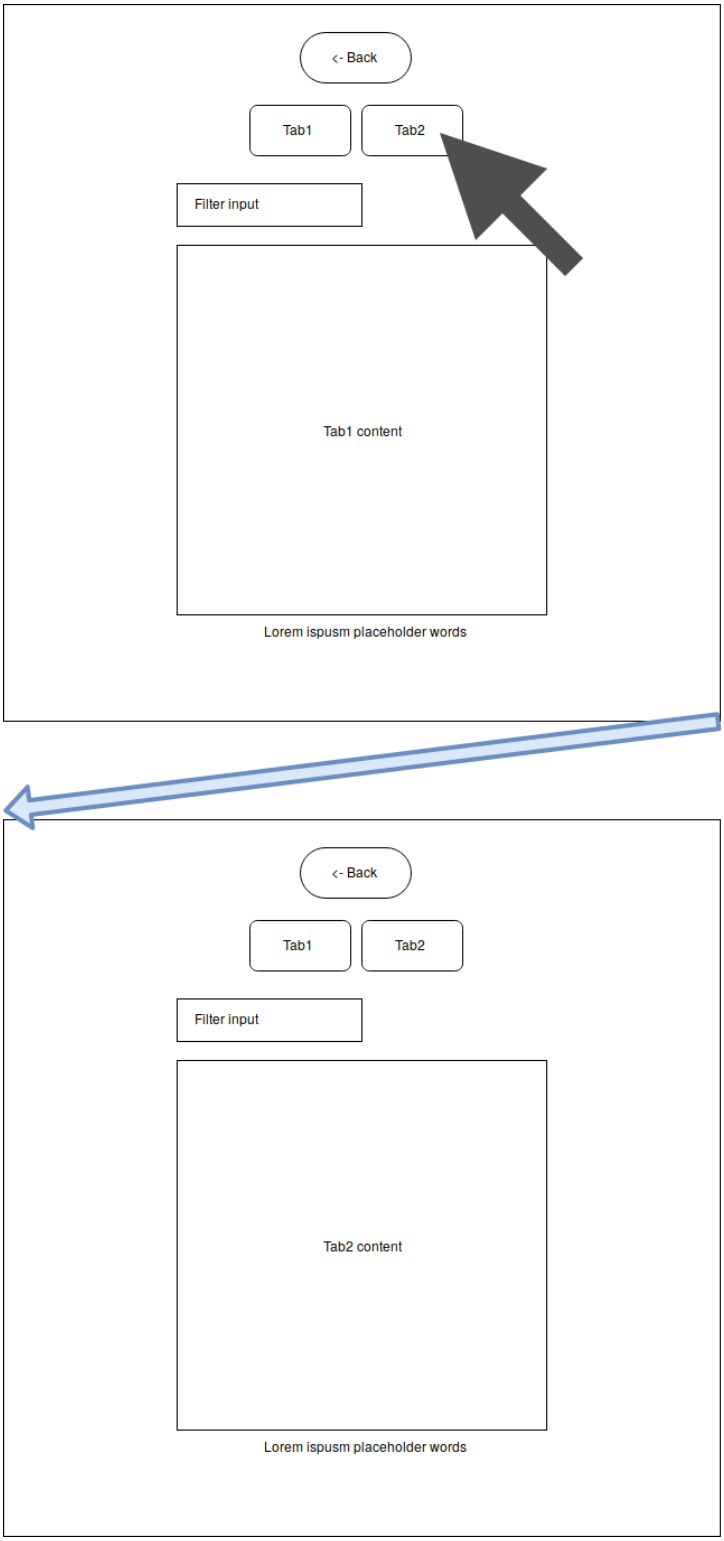


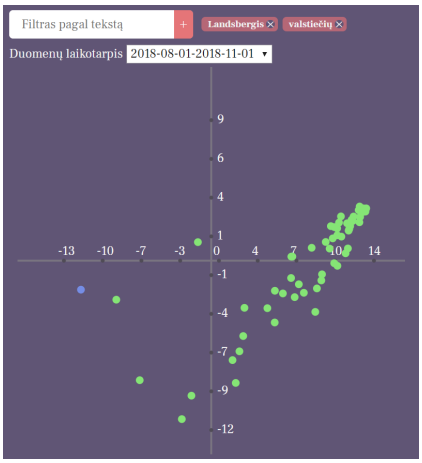
Figure 3: Mockup of atviras-seimas.lt second screen

3.2.2 Scatter plot

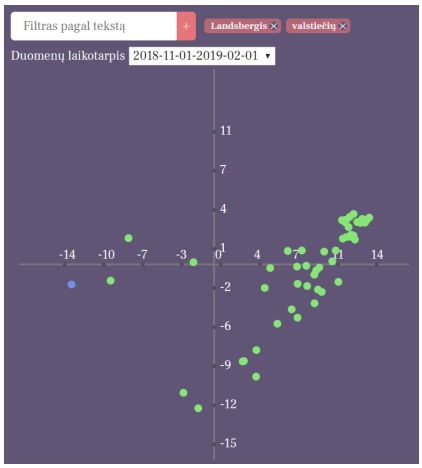
To visualize voting patterns data needs to be projected on two or one dimensional plot. Each dot represented can be different parliament member and its position can indicate its voting pattern. The more scattered two points are - the more different voting patterns are. Each point can have color of faction to easier indicate clusters with human eye. Another important feature is view different time periods, even arbitrarily chosen. This enables users to inspect differences during term of office. Also it is worth to mention that parliament member change factions throughout term of office. For the sake of simplicity point color can be left to be faction color when member was elected to office. This approach has a drawback when looking at data changes during different periods, but it can keep things simplified.

3.2.3 Data filter

For users to inspect data with more than hundred points, filtering for members and factions needs to be present. Users are able to filter data by word. It is possible to filter by one word or phrase, however multiple words or phrases can also be used. In addition, while filtering data users can choose different periods of time. All this helps to compare separate factions to one another or parliament members to factions.



(a) Data filter 1



(b) Data filter 2

Figure 4: Figure 4: Data filters

3.3 Tools

3.3.1 Programming language: *S*cala

Both research and software parts of the project were done using Scala [7]. Scala choice is due its powerful type system and JVM libraries accessibility [8].

3.3.2 Database: *M*ySQL

Since amount of data is relatively small as seen in experiment part - there is no need to consider data distribution problems. In this case the most important factor is to have fewer bugs. To achieve that Structured Query Language (SQL) is picked which has types for each column - MySQL [9].

3.4 API server

This subsystem handles incoming HTTP requests from users. It uses *Akka-http* library [10]. Both computing and application requests are handled in this subsystem. Computing requests tell server to download specific set of data, update to newest data or order server to compute data for scientific methods. Application requests serve data to the client, querying it from the database. Since both client and server side are written with one programming language there are many benefits to use. While client side can remain as a single page application, server side can still share code with it. One such benefit - no need to write serializers and deserializers, since data types can be shared between two projects and process can be automatic.

3.5 Coordinator

3.6 Downloader

Downloader consists of few parts: *Download coordinator, Scraper, Parser, Error handler*.

4 Description of experimental research

4.1 Data statistics

Data statistics are calculated from data LRS open data API which was downloaded and saved to local database. Quite a big chunk data from 1990-2012 is missing due different information systems used at that time and other reasons. It was not migrated to current open data API. For this reason this project focuses more on recent data, but it is not limited to analyze older data too if it becomes available.

Name	Count
Terms of office	8
Sessions	101
Parliament members	1207
Plenaries	3977
Vote rows	3933006
Votes	28121
Agenda questions	64243
Plenary questions (2012-)	15108
Discussion events (2007-)	361996

Table 10: Statistics of downloaded data

4.2 Encoding of data

To use scientific methods data needs to be prepared for them. Two methods: *MDS* and *k-means* need distances between pairs or training rows to compare parliament members to each other. This means that to compare one member's to another some measure like Euclidian distance is needed to produce numerical difference between points. Vote outcomes in given data are categorical, meaning they need to be converted to numbers. Since authors in [3] tried few vote encoding for MDS and LRS older data - this project will use different ones to capture broader scientific results. Data for experiments only includes current term, which start in 2016.

In table 11 chosen voting outcomes encoding can be viewed. *Encoding 1* straightforward scores are chosen. Voting abstain has negative score, because some legislature requires absolute majority to pass, therefore minority of parliament can block by abstaining. Did not vote is considered to be neutral, as some parliament members begin their term of office later and need to be assigned a

score. *Encoding 2* values are chosen more flexibly. Voting for and against are opposites, however voting abstain and not attending are neutral. This is to cover the case where members actually vote not to block specific vote, but don't have opinion on what to vote. *Encoding 3* differs by having negative score for not voting at all. This is to cover the case where half or parliament members plus one is needed to pass certain laws, like changing constitution. This means that by not registering to vote you can actually vote against [11].

Vote outcome	Encoding 1	Encoding 2	Encoding 3
For	2	1	2
Against	-2	-1	-2
Abstain	-1	0	-1
Did not vote	0	0	-1

Table 11: Encoding of vote outcomes

Since factions are important part when viewing visualizations, they need to be consistent throughout graphs. Faction assigned colors can be viewed in figure

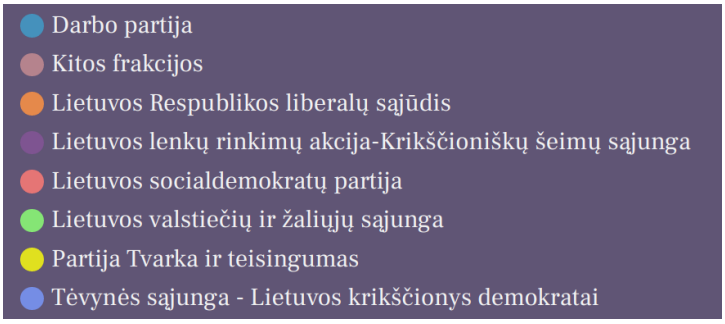


Figure 5: Colors assigned to factions in following visualizations

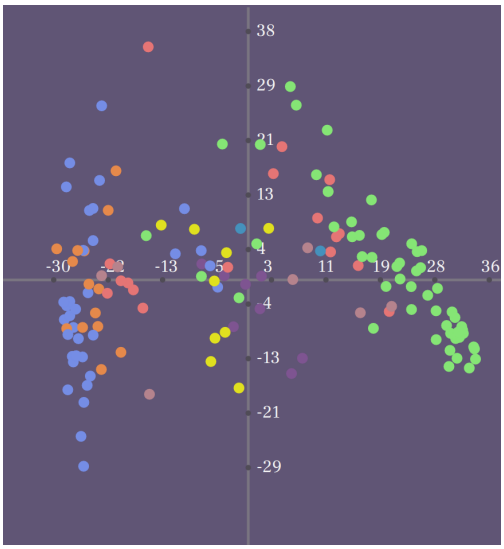
4.3 Multidimensional Scaling

To use MDS method proximity matrix needs to be generated. Distances need to be calculated from given data. Goal is to view voting patterns for given parliament members meaning that connection between somehow encoded votes and parliament members needs to be represented with this matrix.

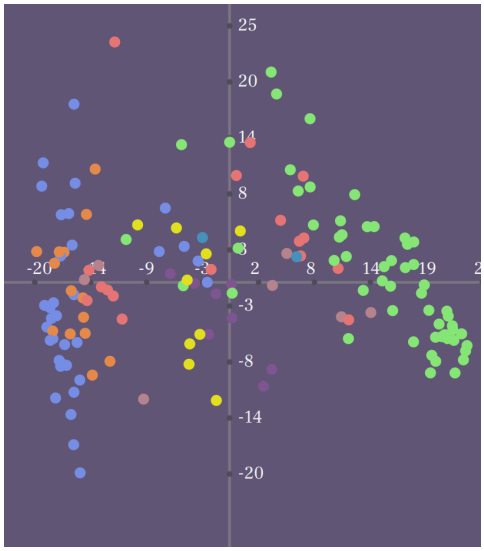
In figures 6a, 6b, 6c MDS for current term of office is calculated and projected on 2d scatter plot. Three variants show how different vote encoding changes plot. Table 12 shows that *E2 encoding* has the highest proportion of variance of the scaled data which was accounted by MDS method. It shows that *E2 encoding* loses the least amount of data, meaning simple vote for and vote against is the best encoding for this case. Also, in visualization, faction voting patterns are clearly visible, as member votes are close to each other. Forming clusters.

Encoding	Proportion of variance
E1	0.752
E2	0.790
E3	0.764

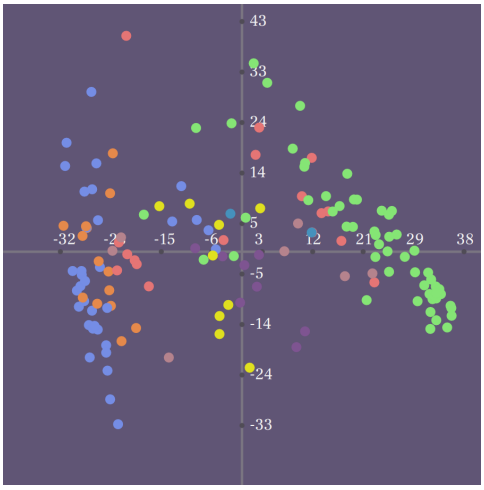
Table 12: Proportion of variance for MDS



(a) encoding = E1



(b) encoding = E2

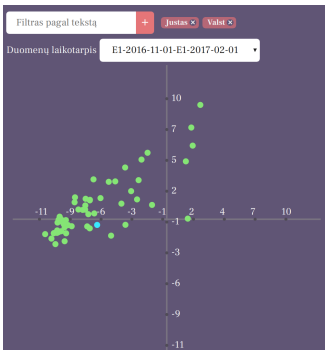


(c) encoding = E3

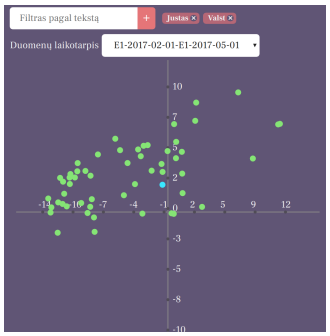
Figure 6: MDS on 2d scatter plot

4.3.1 Time periods

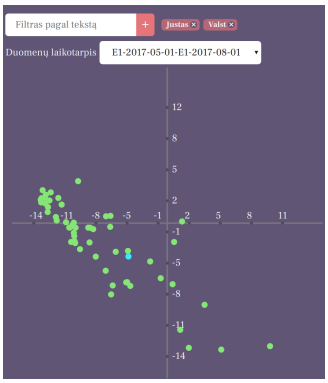
To get more insights, as discussed in problem analysis, data is split into 91.25 day periods. This enables to see what difference each of these periods introduce to the members. For example: Justas Džiugelis was a member of Greens majority faction during 2016-11-14 – 2018-09-10 period. If MDS were plotted to show how his voting is similar to his faction during current term of office - its possible to see how farther he gets before leaving the party. Comparison can be seen in figure 7, where green dots are members from his former faction, and light blue point is him. Before he leave the party on 2018-09-10, his voting patterns start to diverge starting 2018-02-01 voting period.



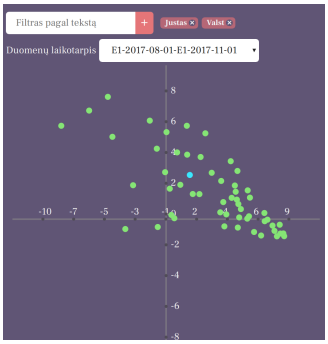
(a) 2016-11-01 — 2017-02-01



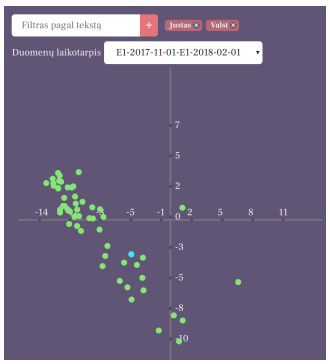
(b) 2017-02-01 — 2017-05-01



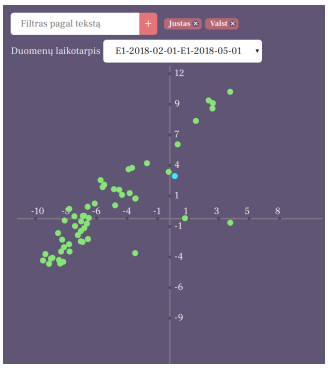
(c) 2017-05-01 — 2017-08-01



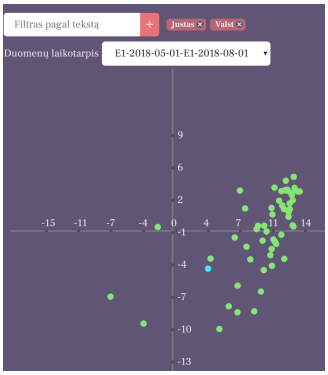
(d) 2017-08-01 — 2017-11-01



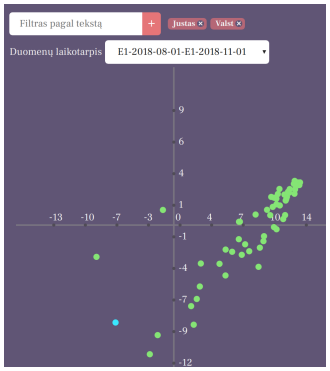
(e) 2017-11-01 — 2018-02-01



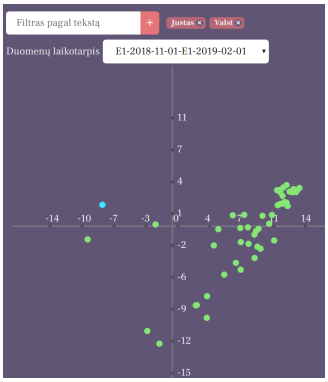
(f) 2018-02-01 — 2018-05-01



(g) 2018-05-01 — 2018-08-01



(h) 2018-08-01 — 2018-11-01



(i) 2018-11-01 — 2019-02-01

Figure 7: Changes in Justas Džiugelis voting patterns

4.4 Unsupervised learning: *k*-means clusterization

There are two goals for *k-means*: classify majority from minority and see if there are groups not explicitly split into factions.

Since some members joined parliament after term of office start or left it before it ended - data is missing on their votes during that time. To avoid this issue - empty training rows are added for that missing time.

Results from *k-means* are displayed on MDS calculated coordinates for the same time period. Training rows for clustering are prepared by encoding votes for given parliament member. Iteration count is 20.

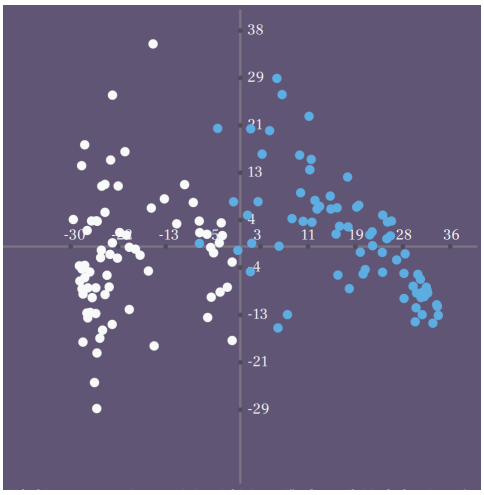
4.4.1 Majority vs minority

Majority and minority are two groups which usually have different ideologies - they couldn't group up, therefore are voting differently on issues. Another experiment could be to test three groups - two for minority and majority and one for outsides, parliament members who left one group or another, changed parties or are voting individually.

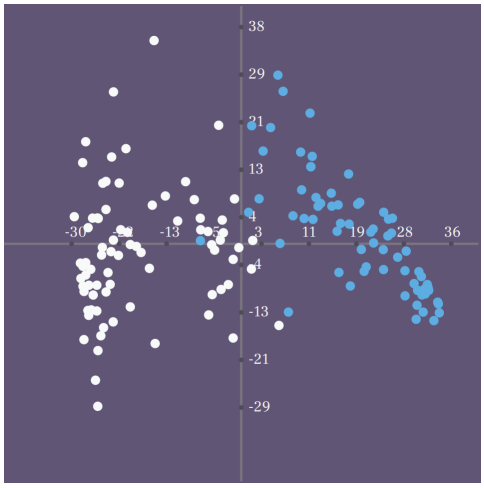
Figures 8a, 8b, 8c can be observed for results projected on MDS coordinates with fixed encoding *E1* for the same time period. Graphs show how parliament members are assigned to two clusters. Just from visual part it is not clear which encoding has best performance. From *K-means distortion* in table 13 shows *E2 encoding* to have the lowest value. Closer look at 8b suggests that most of points are classified correctly in this graph, white points being a minority and blue ones belonging to majority. This finding can also suggest that simple vote for and vote against split for positive negative score is the best as it was for MDS proportion of variance.

Encoding	k=2	k=3
E1	536738	500433
E2	121566	116326
E3	854237	811270

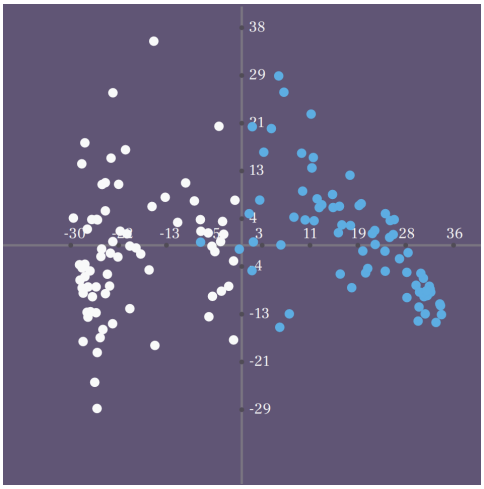
Table 13: Distortion for *k-means* majority vs minority results



(a) k=2, encoding = E1

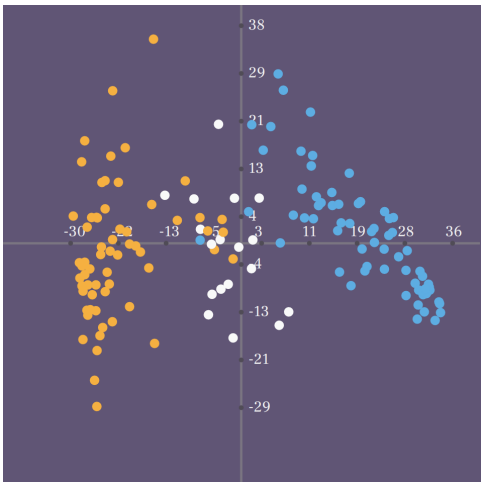


(b) k=2, encoding = E2

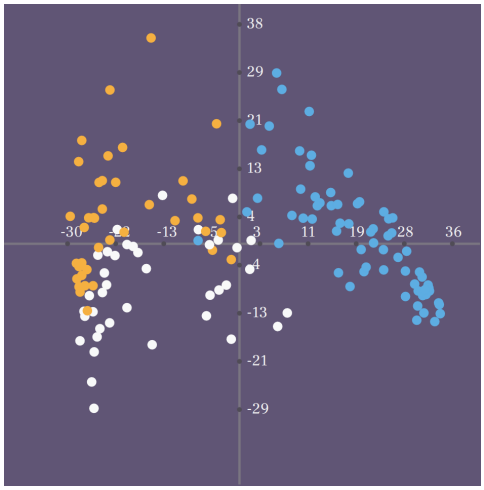


(c) k=2, encoding = E3

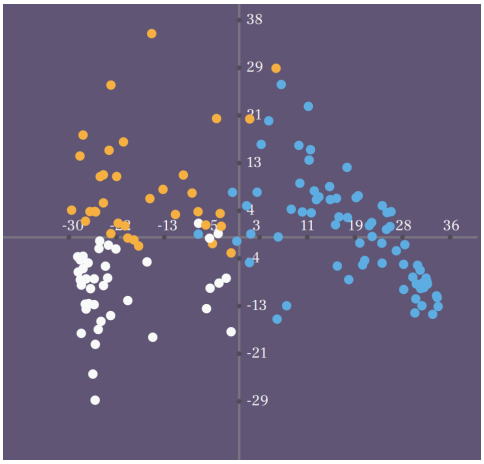
Figure 8: majority vs minority, *k-means* on MDS coordinates



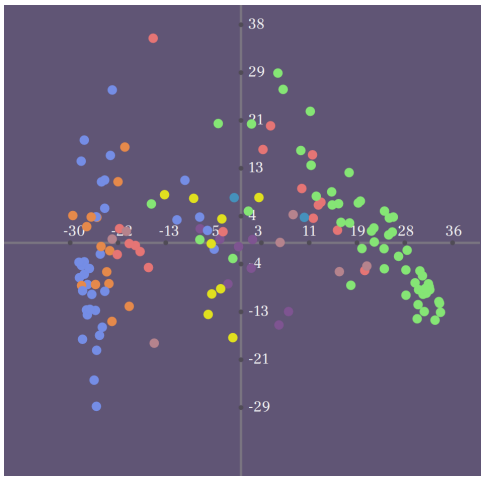
(a) k=3, encoding = E1



(b) k=3, encoding = E2



(c) k=3, encoding = E3



(d) Actual factions

Figure 9: majority vs minority, k=3, *k-means* on MDS coordinates

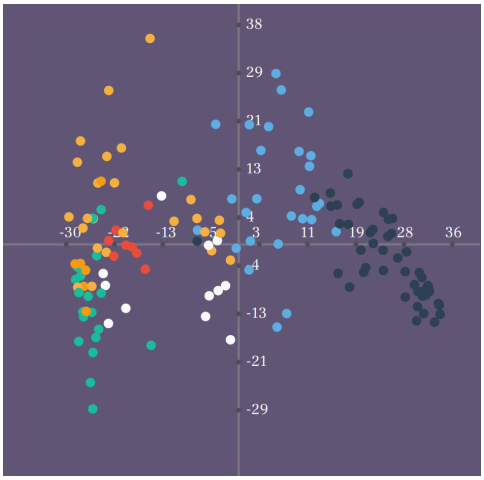
Looking at figures 9a,9b, 9c suggests that *k=3* is one cluster too much for *E2, E3* encodings as it doesn't show anything meaningful. 9a figure looks more shows points which are more truthful, white points being people who are outsiders as predicted in hypothesis.

4.4.2 Secret groups

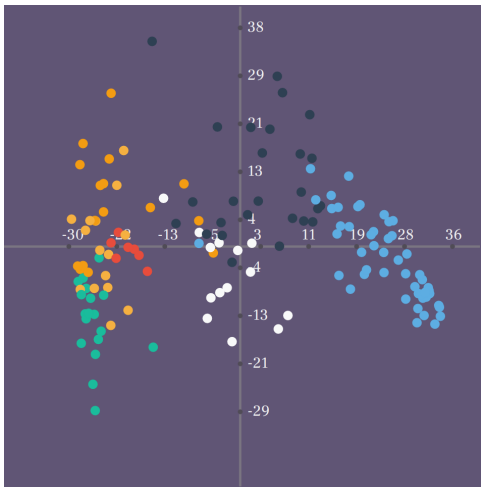
There are 8 factions in current term of office. 8th faction is mixed group, meaning they will not vote in unison. In perfect scenario remaining 7 factions should be classified by classifier. To see how differently voting patterns can group - one group with less than 7 clusters should be tried and another one with more than 8 to see how well it matches current factions.

Encoding	k=7	k=8	k=9
E1	446888	432739	428739
E2	100255	99660	98591
E3	706380	694879	688514

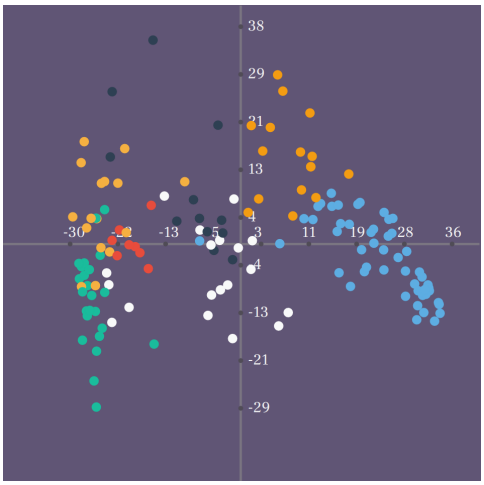
Table 14: Distortion for *k-means* secret groups results



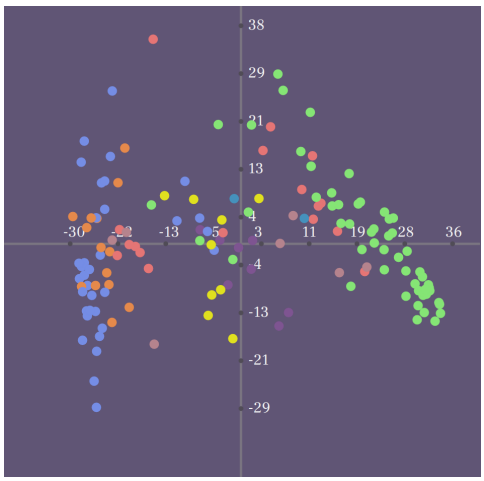
(a) k=7, encoding = E1



(b) k=7, encoding = E2

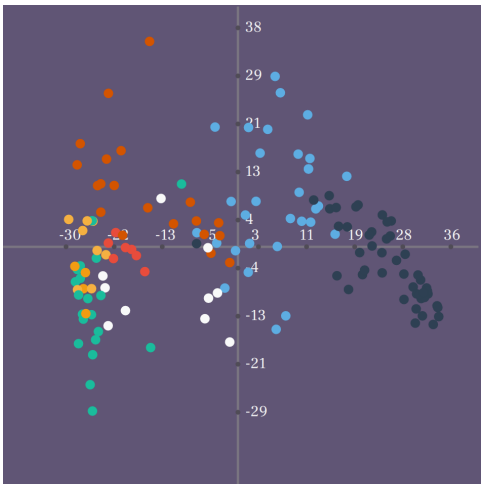


(c) k=7, encoding = E3

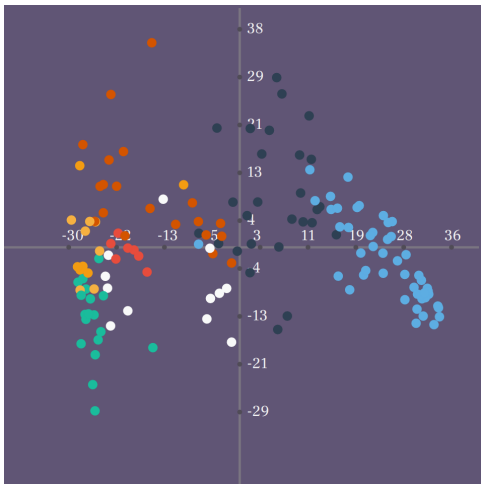


(d) Actual factions

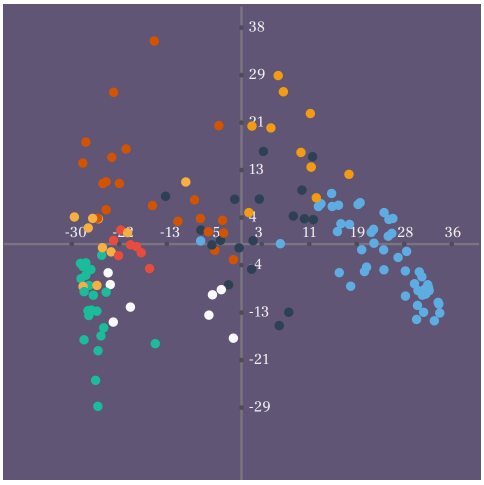
Figure 10: different clusters of members, *k-means* on MDS coordinates



(a) k=8, encoding = E1

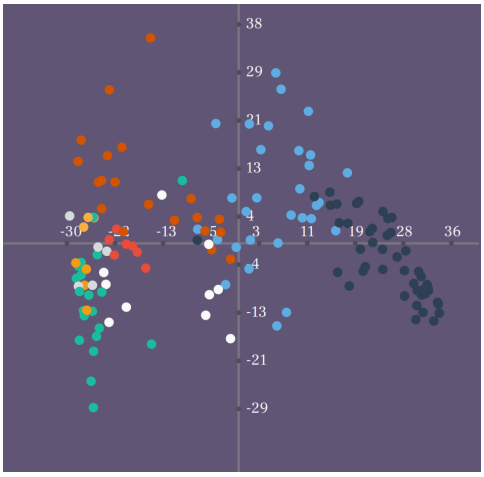


(b) k=8, encoding = E2

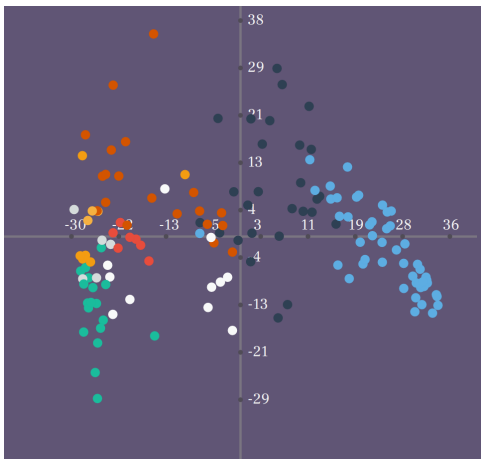


(c) k=8, encoding = E3

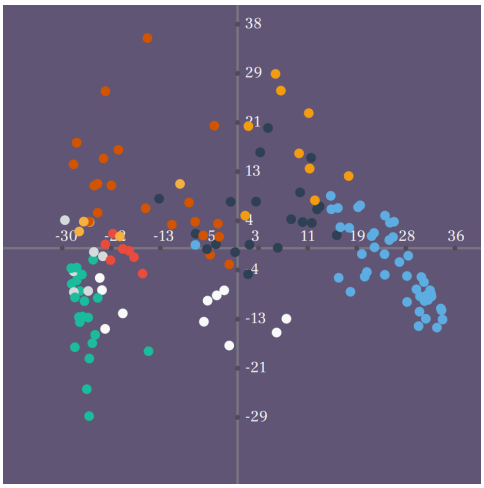
Figure 11: different clusters of members, *k-means* on MDS coordinates



(a) k=9, encoding = E1



(b) k=9, encoding = E2



(c) k=9, encoding = E3

Figure 12: different clusters of members, *k-means* on MDS coordinates

4.5 Results and conclusions

References

[1] K. T. Poole, *Spatial Models of Parliamentary Voting*. Analytical Methods for Social Research, Cambridge University Press, 2005.

[2] T. Krilavicius and A. Zilinskas, "On structural analysis of parliamentarian voting data," *Informatica, Lith. Acad. Sci.*, vol. 19, no. 3, pp. 377–390, 2008.

[3] V. Mickevičius.

[4] A. Boche, J. B. Lewis, A. Rudkin, and L. Sonnet, "The new voteview.com: preserving and continuing keith poole's infrastructure for scholars, students and observers of congress," *Public Choice*, vol. 176, 05 2018.

[5] I. Borg and P. Groenen, *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.

[6] D. T. Pham, S. S. Dimov, and C. D. Nguyen, "Selection of k in k-means clustering," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 219, no. 1, pp. 103–119, 2005.

[7] "Scala programming language." `https://www.scala-lang.org/`. Accessed: 2019-05-06.

[8] J. Suereth, *Scala In Depth*. 2012.

[9] "Mysql documentation." `https://dev.mysql.com/doc/`. Accessed: 2019-05-06.

[10] "Akka-http." `https://doc.akka.io/docs/akka-http/current/index.html`. Accessed: 2019-05-06.

[11] "Lietuvos respublikos konstitucija." `https://www3.lrs.lt/home/Konstitucija/Konstitucija.htm`. Accessed: 2019-05-06.

A Appendix