DATA 501: Final Project Report

Submitted to:
Dr. Irene Vrbik
Irving K. Barber School of Arts and Sciences
Department of Computer Science
University of British Columbia
Kelowna, Canada
irene.vrbik@ubc.ca

Submitted by:
Mohammad Abdul Hadi
Student ID: 99892168
Irving K. Barber School of Arts and Sciences
Department of Computer Science
University of British Columbia
hadi@alumni.ubc.ca

ABSTRACT

An interactive geo-spatial data visualization tool has been developed to represent and interpret the party-region-wise outcome of the Canadian Federal and Provincial Elections which have already taken place at different times throughout the history. A huge number of databases containing a lot of facts and numbers are already being maintained by numerous organizations over the world but none of them offer any concise and insightful data interpreting or visualization tool which can help user to process all the information in a conveniently compact and consistently comparable way. To remedy the issue, this tool has been designed to scrape all the available data from particular database and convert them into visual information for user convenience. Apart from the geo-spatial mapping, I leveraged other popular data visualization techniques in order to put all the available information together in a well-knitted manner. On top of that, this tool can also perform usual trend analysis, forecasting with decent accuracy using typical Machine Learning techniques such as regression line and gradient descent. Altogether, this visualization tool enables any user to gather considerable amount of knowledge about the details and specifics of past federal, territorial elections as well as help them to predict the certain factors (i.e. number of candidates, chance of winning) concerning the future election with moderate preciseness.

Keywords: geo-spatial map; Canada-map; election-visualization; election-database-scraper; r-mapping; tidyverse-map; .shp-file-manipulation;

INTRODUCTION

The 2019 Canadian federal election (formally the 43rd Canadian general election) was scheduled to take place on October 21, 2019, to elect members of the House of Commons to the 43rd Canadian Parliament. As a part of the election campaign, I was handed out a considerable number of pamphlets and leaflets. But as a newcomer in this country, I

felt the necessity to know about the outcomes and details of the past elections. While a huge amount of data is available and being maintained by the government in form of web-based databases, I have found the concise, insightful data-interpretation using adequate visualization techniques to be lacking. Elections data has many applications; we can analyze the data and search for trends by looking at party support over time, or the geographic trends of party support for any given election. I think it would be convenient to have a tool to explore the huge amount of election data in the most efficient way where all correlated factors like "Seats Won by Political Parties", "Vote Share by Political Parties", "Seats Share by Political Parties" on any given election would be categorically and graphically represented. That's why, I intended to design an interactive platform which will provide a geo-spatial visualization of Canadian territories, or provinces and some info-graphs/charts which can improve user cognition by utilizing graphs in order to see patterns and trends for the participating parties. In the geo-spatial map of Canada, the provinces will be color-coded by their choice of parties for any selected option from all the available federal and provincial election years since 1867. For the election data, I relied on a web-embedded database created by The School of Public Policy of University of Calgary where we can get all the relevant data for all the elections since 1867. To scrape the database efficiently, I have designed an extensive Python scraper. For geo-spatial plotting of the map, I used an R-script where numerous useful libraries have been utilized. Tableau was used to formulate the infographic charts.

MATERIALS AND METHODS

The whole project consists of 3 different components. These components required different technologies and programs to accomplish the given job. Three sections are enlisted as below to best-explain the function of the different components:

i. *A python scraper* has been designed to efficiently scrape through the web-embedded database and conveniently store all the information in different comma separated values in exactly the way other components of the project require them to be.

ii. *An R script* was written integrating some packages from some useful libraries to visualize the geo-spatial map of Canada with proper representation of the election data that was gathered by the scraper.

iii. *Tableau* was used to generate some useful graphs for the user to better comprehend the information, forecasting and analyze the trend for the upcoming elections. Options were provided for the users so that each user have a choice to see different

representation of the same information using different graph-types (i.e. horizontal/ vertical bar-chart, pie/donut-chart, heat-map of different colors).

These 3 components have been described below in details.

i. Python Scraper

For this project, we required valid dataset of Canadian elections which were collected and validated by well-reputed source. For the election related data, the database developed by Dr. Anthony Sayers, Department of Political Science, University of Calgary is one of the most reliable, consistent, and popular databases. Link to the database is given here:
http://canadianelectionsdatabase.ca/
This database contains information on federal, provincial, and territorial elections since 1867 arranged by election, party, candidate and district. The database allow user to explore the data in myriad ways. I chose this database as I found the website engaging which contains around four million data points.[1]
But unfortunately, the developers of the database did not provide any downloadable option so that other people can retrieve and use this information for varying purpose. So, I decided to develop a scraper which will scrape, extract necessary data and store them in appropriate format which will be useful and accessible to other components of this project. Storing data in specific fashion can reduce the need for further data-cleaning and polishing. Storing data as it is found in the database may lead to exhaustive data-formatting for the reusability purpose.
We chose Python for the scraper as it provides a powerful, robust package for the web-crawling and data scraping, named "Beautiful Soup". Beautiful Soup is a Python package for parsing HTML and XML documents (including having malformed markup, i.e. non-closed tags, so named after tag soup). It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping.

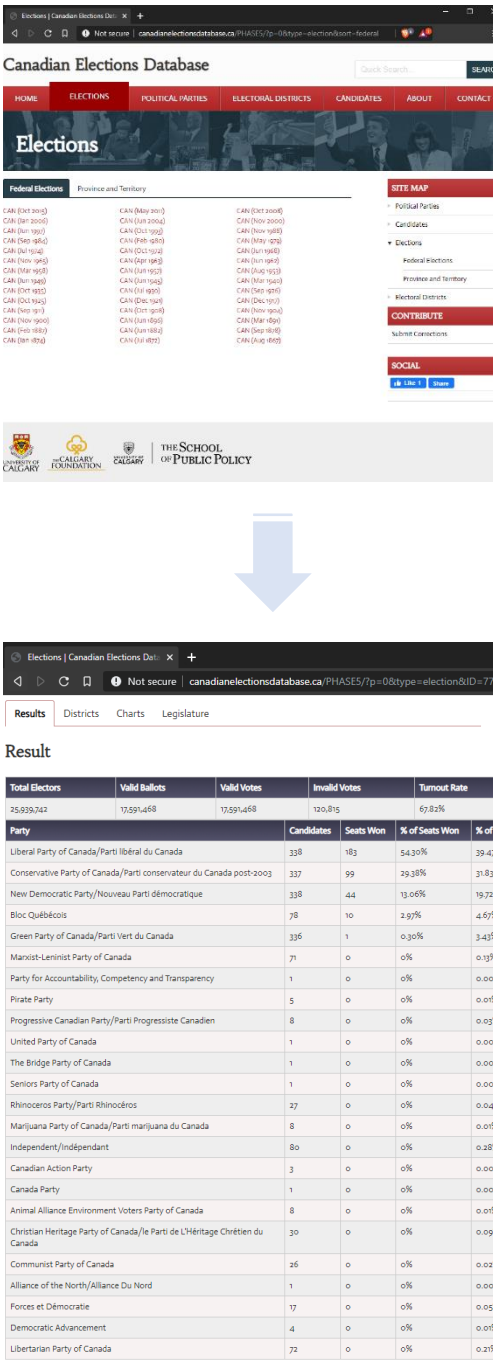


Fig 1.1: View of web-embedded Database (static pages)

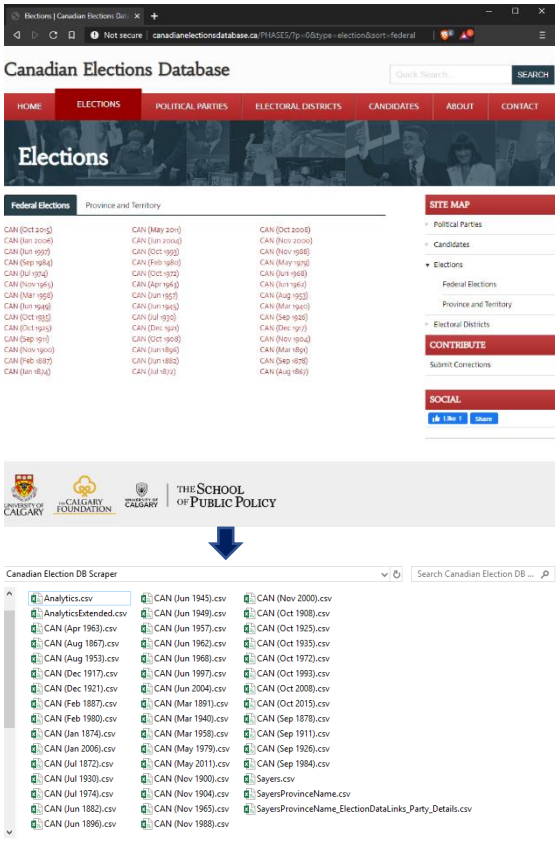
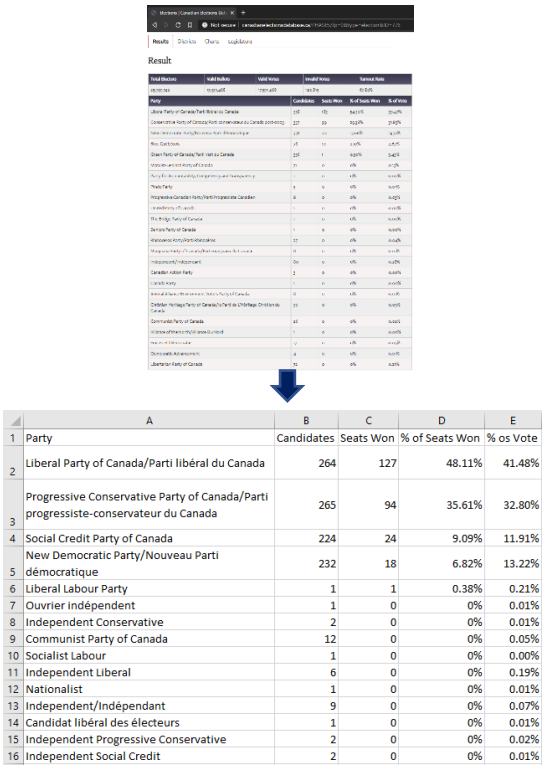


Fig 1.2: Conversion of web-embedded Database (static pages[up]; dynamic pages[down])



	A	B	C	D	E
1	Party	Candidates	Seats Won	% of Seats Won	% os Vote
2	Liberal Party of Canada/Parti libéral du Canada	264	127	48.11%	41.48%
3	Progressive Conservative Party of Canada/Parti progressiste-conservateur du Canada	265	94	35.61%	32.80%
4	Social Credit Party of Canada	224	24	9.09%	11.91%
5	New Democratic Party/Nouveau Parti démocratique	232	18	6.82%	13.22%
6	Liberal Labour Party	1	1	0.38%	0.21%
7	Ouvrier indépendent	1	0	0%	0.01%
8	Independent Conservative	2	0	0%	0.01%
9	Communist Party of Canada	12	0	0%	0.05%
10	Socialist Labour	1	0	0%	0.00%
11	Independent Liberal	6	0	0%	0.19%
12	Nationalist	1	0	0%	0.01%
13	Independent/Indépendant	9	0	0%	0.07%
14	Candidat libéral des électeurs	1	0	0%	0.01%
15	Independent Progressive Conservative	2	0	0%	0.02%
16	Independent Social Credit	2	0	0%	0.01%

The Beautiful Soup package has some effective methods which return/download the HTML page upon sending a request to the server with corresponding URL. From the downloaded HTML page, we can find the element which contain the necessary information such as <table>. Specific element was returned through searching their designated class or id. Once we locate the required element, we parse the data into a list to store the list later into a csv file. The intermediate <list> helps to modify huge chunk of data as per the requirement of other components in the project. For some cases, same information was saved in different .csv files for convenience and reusability.

A particular challenge appeared when I discovered that some pages were communicating with the database using AJAX and JS requests. Beautiful Soup does not provide support for such scenarios. I had to thoroughly inspect the pages to dig out specific requests which would in turn return the data we require. Such two files were *request.controller.php* and *fetchData.controller.php*.

To retrieve the data that were being requested by AJAX or JS request, I had to listen in the port that communicates with the database and download the database response. To accomplish this task, I utilized another powerful, robust package "Selenium" which enabled me to capture both AJAX and JS responses.

Although the method I used was highly dependent on *connection_time.* If the response did not appear within a specified time period, the method terminates the listening action. So, I put the method into a *try_catch* block and made the *connection_time* to be an incrementing variable. So, for a specified *connection_time,* if the method could not capture a request, it would try again with an increased *connection_time.*

For atomizing my tasks, I have separated my codes into two different programs. One works on the static pages and the other captures and stores the AJAX and JS requests. The scraper-programs have been uploaded in the GitHub. Link is being attached:

https://github.com/Mohammad-Abdul-Hadi/scraper-for-canadianelectionsdatabase.ca

ii. Geo-spatial map visualization using R

For this part of the project, I tried to build a map of Canada and integrate election information into it. This is pièce de résistance of my project. The gist of the process is grabbing a shapefile and converting it to simple-features object for use in R.

	A	B	C
1	PRUID	Province	Party
2	10	Newfoundland and Labrador / Terre-Neuve-et-Labrador	Liberal Party of Newfoundland and Labrador
3	11	Prince Edward Island / Île-du-Prince-Édouard	Progressive Conservative Party of Prince Edward Island
4	12	Nova Scotia / Nouvelle-Écosse	Nova Scotia Liberal Party
5	13	New Brunswick / Nouveau-Brunswick	Progressive Conservative Party of New Brunswick
6	24	Quebec / Québec	Coalition Avenir Québec - L'équipe François Legault
7	35	Ontario	Progressive Conservative Party of Ontario
8	46	Manitoba	Progressive Conservative Party of Manitoba
9	47	Saskatchewan	Saskatchewan Party
10	48	Alberta	United Conservative Party
11	59	British Columbia / Colombie-Britannique	British Columbia Liberal Party
12	60	Yukon	Yukon Party
13	61	Northwest Territories / Territoires du Nord-Ouest	Sans Nom/ No Name
14	62	Nunavut	Nunavut Independent

Fig 2.1

We used tidyverse as our core package to convert map data into plot. Converting map data into a format that R packages can methodically use requires a bit heftier lifting behind the scenes than the core tidyverse libraries provide. We needed to install the sf library, which conveniently brought several useful dependencies with it. Some other packages were loaded which had provided functionalities that we had to use in the conversion and mapping process. Installed packages are listed below:

```
install.packages("sf")
install.packages("rgdal")
install.packages("geojsonio")
install.packages("spdplyr")
install.packages("rmapshaper")
```

Fig 2.2

I have built a function, *theme_map()*, as *ggplot* theme. It does nothing but turning off insignificant pieces of the plot (so that it looks neat) that we don't essentially require.

Next, we need to get the actual map data for developing Canadian map. This was the hardest bit. We were fortunate that Canadian Central Statistics Agency provided map shape files that we can use. The Shape File format (.shp) is the most popular and widely-used standard format for map-data.

So, I got the data from CSA's well-organized website. Link is provided: http://www12.statcan.gc.ca/census-recensement/2011/geo/bound-limit/bound-limit-2011-eng.cfm.

Fig 2.3

I have decided to use *ArcGIS .shp file*; although there were other available options. But GIS (Geographical Information System) seemed to be much simpler than the others (i.e. Geography Markup Language (.gml) or MapInfo (.tab)). I have chosen the category *Census divisions* and *cartographic boundaries* as they would help me during the integration of the election data.

I have imported the shapefile using one of *rgdal*'s functions, *readOGR*. Then I converted the returned object to *GeoJSON* format and simplified the polygons. These steps had taken a long time to run. The resulting *GeoJSON* file was saved, which was ready for further direct implementation.

Importing, Converting, and Thinning should only be done once. The resulting data was saved in the data folder within the root directory. *GeoJSON* file became the building block for further coding.

After that, we read the *GeoJSON* file back in as an *sf* (simple features) object. As discussed earlier, the *.geojson* file is also included in the repository, so for the test purposes one can load the data and start working from here-on-out.

Fig 2.4

I have rigorously checked the metadata and found that the projection-type was not globally-accepted; so, I changed that to the Lambert Conformal Conic projection.

Finally, and had the election data at my hand which my python scraper stored as census-district-level data and province-territory-level-data. Now we needed to merge this data onto the map-data produced as *canada-cd*. We need to remember that *canada-cd is just a* big data frame specifying a large number of lines that need to be drawn in the plot. Now we merge our election-dataset with that data frame.

This merge step is significant as we must take care that the values of the key variables we are matching on, really do exactly correspond to one another. Otherwise, missing values will be introduced into data-frame, and the lines on the map will not smoothly join up. This will result in a shredded map as R tries automatically fill the missing portion of the polygons.

For out dataset, the PRNAME variable is the only attribute with the same name in both the data sets we are joining, but its too risky to join two datasets using character attributed column. An example is provided to illustrate the issue. For example, if rows corresponding to British Columbia/Colombie britannique, were named as it is mentioned in the region variable of election data frame but "British Columbia (BC)" in the corresponding region variable of map data, then merging on PRNAME would mean introducing NA values in the map-data frame. Broken Maps are usually caused by these kind of merge errors. One other example can be, one of our province names inadvertently has a leading or trailing space as a result of the data originally being imported from web-embedded database. That would mean, for example, that "Alberta " and "Alberta" are different strings, and the match would fail. That's why joining two datasets using character attributed column is highly discouraged.

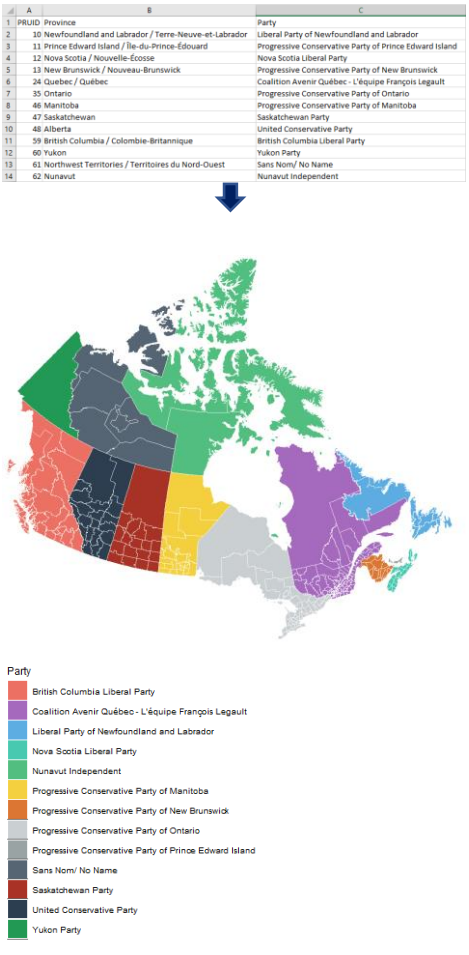


Fig 2.5

So we have chosen PRUID to merge our data-frames. For the data scraped from database, the province name did not have PRUID attached to them. But we can remedy the fact by using simple Excel formula and introduce new column PRUID (which store the corresponding ID for the province appeared in that row) as shown in Figure 2.1. Before merging the data, I pruned some unnecessary columns and converted the *factor* attributed columns to *numeric* for data-compatibility during *join.*

```
canada_mp$CDNAME <- NULL
canada_mp$CDTYPE <- NULL
canada_mp$PRNAME <- NULL
canada_mp$PRUID <-
as.numeric(levels(canada_mp$PRUID))[canada_mp$PRUID]
canada_mp$CDUID <-
as.numeric(levels(canada_mp$CDUID))[canada_mp$CDUID]
```

After loading the election data, we convert its' *sf_ tbl* type and read it into *data_ frame* before proceeding any further. We then use *left_ join()* function. R smartly identify similar column on which it should perform the join by reading the column names and attributes. Although if columns have different names in each data set, we could specify that if needed.

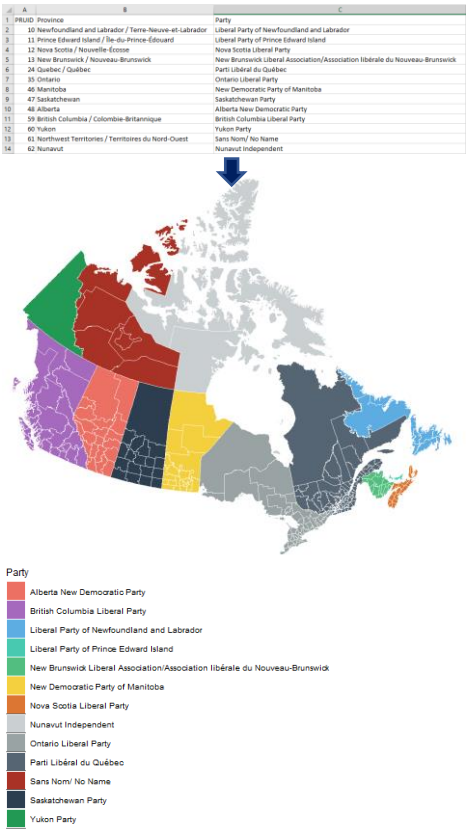


Fig 2.6

I used a vector of repeated colors to fill in the map, for decoration only. Before plotting the map data, we need to convert the *numeric* columns back to *factors.* Otherwise it would result in an error saying that *discrete data have been provided instead of required continuous data.*

```
canada_mp$PRUID <- as.factor(canada_mp$PRUID)
canada_mp$CDUID <- as.factor(canada_mp$CDUID)
canada_mp$Province <- as.factor(canada_mp$Province)
canada_mp$Party <- as.factor(canada_mp$Party)
```

Then we just plot the map filling the colors using "party" attributes to color the regions according to their elected parties. A self-contained R project with this code is available on GitHub. Link is given below: https://github.com/Mohammad-Abdul-Hadi/Canada-Map-R-Visualization-for-Election-Data

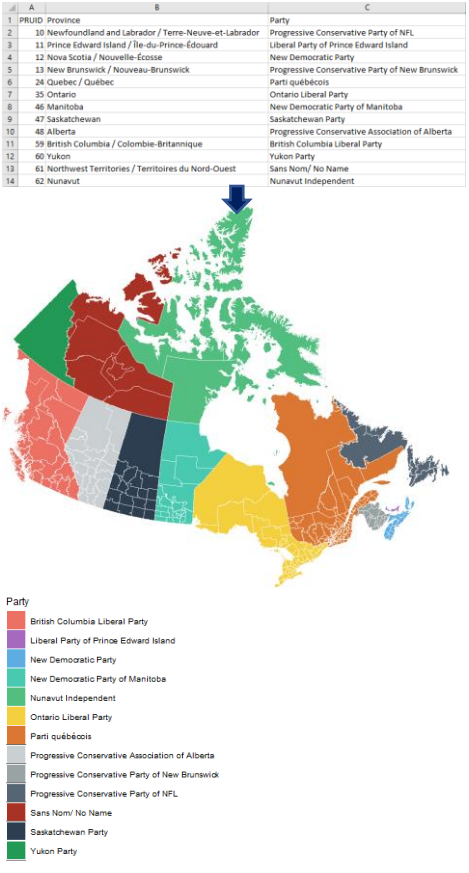


Fig 2.7

Fig 2.2 shows the territorial borders of Canada. This was the most initial stage of mapping. No color-coding was implemented.

Fig 2.3 and 2.4 shows the coloring of the map according to PRUID and CDUID (census division) respectively. Fig 2.5, 2.6, and 2.7 shows how particular election data can be converted into the map. Fig 2.5, 2.6 represent the provincial elections and the latter one (Fig 2.7) represent the federal election.

Please note that all the election data (on an average: 31 elections from each of the province (13 province) can be represented using the R-script). To avoid redundancy, I just demonstrated 3 election outcomes. Also, these 3 samples were chosen randomly from over 400 such maps generated. We can save all the maps into our directory using the following code segment:

```
ggsave("D:/Area 51.2/UBC/Semester 1/DATA 501/Canada-Map-R-
Visualization-for-Election-Data/FirstProject/figures/Canada-
election-1865.jpg", p_out, height = 12, width = 15)
```

iii. Tableau for graph-development

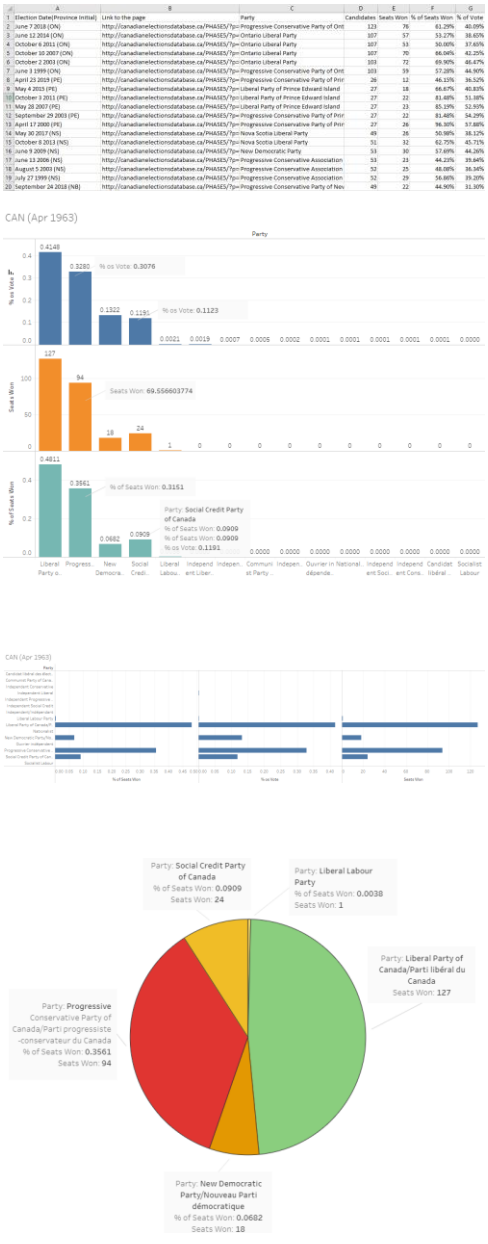


Fig 3.1

For this final part of the project, I tried to mitigate the limitations that came hand-in-hand with map-visualization. Not too many information can be attached in the map as it would make the map messy. At the same time to know which party won in which election in a neat way was not the entire goal of my project. I wanted to give users a concise representation of the whole picture as well as the forecasting power for analyzing upcoming-election. It can also help people to extrapolate and make educated guess about the election outcome; both provincial and Federal. That's where Tableau comes into play. It helps me to convert my data into robust graphs which can give user some useful insights. Moreover, user would get a chance to represent their data through any sort of graph of their choice. Let's delve into it.

An illustration of how user can choose from different graphing techniques is provided in Fig 3.1. Here Canadian Election data of 1963 have been represented using 3 different graphs.

a) Vertical Bar Chart
b) Horizontal Bar chart
c) Pie chart

There are other graphs generated from where user can choose to represent the data such as: donut chart, heat map etc. The idea is to provide users maximum comfort and options while they explore the huge range of data. At the same time, we need to remember that too many options can be proved to be detrimental to providing maximum comfort for the user. Please note that, for any given election, user has the independence to generate any of the mentioned graph. To avoid redundancy, we provide only 3 graph forms for a randomly selected election. I hope this much would suffice for report's illustration.

The next illustration Fig 3.2 shows the total candidate numbers in the federal elections since 1867. We can see that number of candidates are ever increasing. I have implemented regression line method to moderately predict number of candidates in 2019 election. Prediction of the illustration seems to be pretty close to the original result. Here, median and average of the number of candidates were also shown. For any of the 42 elections, if user hover over the line the details will be shown in text pop-ups. For future prediction, user would have to hover over any point on the best-fit line.

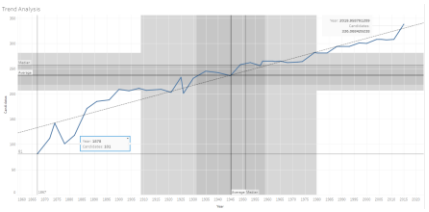


Fig 3.2

The illustration in Fig 3.3 shows the federal parties' history of winning in the past elections. I have chosen heat-map as it is the most popular to consistently compare the volumetric and color intensity difference with minimal effort. Human cognition automatically picks up the information that in the history of Canada, "Liberal Party" has won the most election. User can also pick that the second and third most popular parties in Canada always went neck-to-neck when it comes to winning federal elections. Although the user may need to possess certain knowledge about the party-names and which parties are still into play. But user can gain this knowledge from other graphs that are involved in this project.

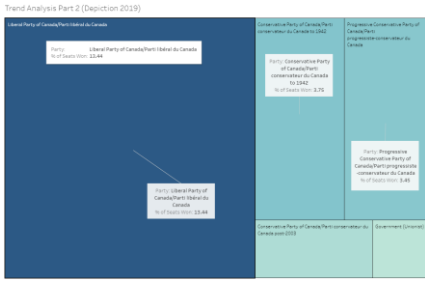


Fig 3.3

In Fig 3.4, a representation is shown where all parties' several information regarding all federal elections were put together such as ("# of seats won", "% of seats won", "% of votes won") along with the regression lines generated for each party. This graph will help user to identify which party has been consistent over the year, which party has been trending over past couple of elections, which parties' popularity (a feature that is depicted by other attributes like percentage of seats and votes won). From this illustration, we can see that although Liberal Party has won most of the elections throughout history, its' popularity has been slowly decreasing over the long run whereas Conservative Party of Canada is gaining tremendous support in recent decades, and Unionists have been holding their position steadily throughout the history.

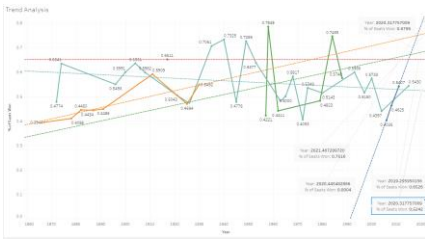


Fig 3.4

Fig 3.5 and 3.6 deals with provincial elections unlike the other graphs. Here all the provincial parties participated to demonstrate which ones gained most of the votes throughout the history. The same information has been interpreted using the heat-map in Fig 3.6. Here too, user has the independence to choose from any representation the may like.

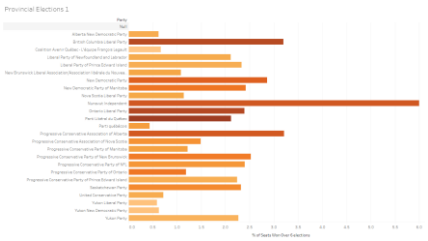


Fig 3.5

Unfortunately, I have used the 10 days online trial of tableau which does not permit me to download my code. Hence, I am attaching the screenshots. Tableau public Workbook Link:

https://10ay.online.tableau.com/t/data501/authoring/DATA501/Sheet1#1
https://10ay.online.tableau.com/t/data501/authoring/DATA501/Sheet2#1
https://10ay.online.tableau.com/t/data501/authoring/DATA501/Sheet2#1
https://10ay.online.tableau.com/t/data501/authoring/DATA501/Sheet2#1
https://10ay.online.tableau.com/t/data501/authoring/DATA501/Sheet2#1
https://10ay.online.tableau.com/t/data501/authoring/DATA501/Sheet2#1
https://10ay.online.tableau.com/t/data501/authoring/DATA501/Sheet2#1

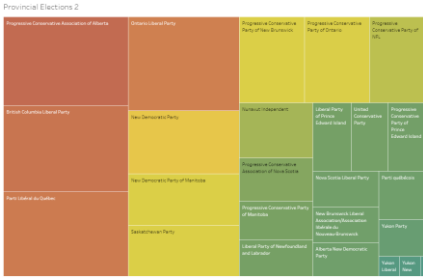


Fig 3.6

RESULTS

My main goal for this project was to introduce the user to a data visualization tool which will interpret the Canadian election data in a sensible, comprehensible, reproducible way using different *ideas* and the *methods* of Data Visualization. I have received positive feedback when I let some users use my tool for gaining information about Canadian election. A small sample-population (14 people) from the grad and under-grad students were chosen at random as users (there were 6 Canadian and 8 international students). 6 of these international students had almost no knowledge about Canadian Election. They were asked to use the tool for 10 minutes and to surf around the database for another 10 minutes. All of them reported that the visualization tool was far more effective for knowledge gathering, comprehension and comparison. All 14 of the users preferred the visualization tool to the raw data provided in the web-embedded database.

REFERENCES

[1] Sayers, A.M. (2017). Canadian Elections Database. Retrieved 1 July 2017, from *http://canadianelectionsdatabase.ca*

[2] Beautiful Soup (HTML parser) In *Wikipedia*. Retrieved December 1, 2019, from *https://en.wikipedia.org/wiki/Beautiful_Soup_(HTML_parser)*

[3] Healy. K. (2019). Data Visualization: A Practical Introduction (1st edition). Princeton, New Jersey.
Princeton University Press

[4] R Documentation and Manuals (2019) Retrieved December 1, 2019 form https://www.rdocumentation.org/

[5] R packages for data science (2018) Retrieved December 1, 2019 form https://www.tidyverse.org/

[6] selenium 3.141.0 (2019) Retrieved December 1, 2019 form https://pypi.org/project/selenium/

[7] Healy. K. (2019). R-bloggers: Geo-spatial map. Retrieved December 1, 2019 form https://www.r-bloggers.com/canada-map/

[8] Statistics Canada (2011). 2011 Census-Boundary files. Retrieved December 1, '19 form https://www12.statcan.gc.ca/census-recensement/2011/geo/bound-limit/bound-limit-2011-eng.cfm

[9] Shapefiles (2019) Create your own shapefiles. Retrieved December 1, 2019 form https://wiki.openstreetmap.org/wiki/Shapefiles