Estimating Slovak Parliamentary Election Results from Partially Counted Data

Juraj Hledik

January 23, 2022

ABSTRACT

We develop a statistical model to estimate election results from partially counted data, i.e. from data available during the actual vote counting. We combine the results from the last Slovak parliamentary elections with location data on Slovak towns. An equilibrium network model where towns' expected vote count and alignment is influenced by its neighbors is used in the process. We also provide a naive probabilistic view of the final results and their associated fair betting odds. Our emphasis is put on simplicity and the speed of calculation such that the framework can be easily and quickly applied during an ongoing future vote count, for instance for live betting purposes.

1 Introduction

Predicting the outcome of an election

2 Data

Two different architectures are used for data import. Firstly, we use two datasets that are stored locally in the project folder. Secondly, we are using data downloaded from a google spreadsheet via the google sheets api functionality. Each method shall have its own subsection here.

2.1 Offline data

We are using two different offline data sets in this project - a voting data from 2020 Slovak parliamentary election and a location dataset of Slovak towns. In Slovakia, there are 5 different inclusion levels to which each voting center is allocated. The top level is called "kraj", then it's "obvod", "okres", "obec" and finally "okrsok". We shall translate "obec" as "town", otherwise we keep the original terms [1]. The numbers of the respective levels is shown in Table 1. The structure of these data sets can be found in Table 2 and 3. The data can be found in the subfolder /data in files historical_data.xlsx and towns.csv.

level	Count
kraj	9
obvod	50
okres	80
town	2817
okrsok	5998

Table 1. Geo-spatial levels.

In Slovakia, results for a given okrsok are reported at once and submitted to the system. In other words, an okrsok is the lowest level of granularity for which the data is observable.

[1] One might try to translate these terms to "states", "counties", "precincts", etc, but the end result would ultimately be simply counter-productive and just overly confusing

town_code	Code of the relevant town
latitude	Latitude of the relevant town
longitude	Longitude of the relevant town

Table 2. Variables available regarding town location.

kraj_code	ID of the relevant "kraj".
kraj	Name of the relevant "kraj".
obvod_code	ID of the relevant "obvod".
obvod	Name of the relevant "obvod".
okres_code	ID of the relevant "okres".
okres	Name of the relevant "okres".
town_code	ID of the relevant "town".
town	Name of the relevant "town".
okrsok	ID of the relevant "okrsok".
party_ID	Unique identifier of the party.
party_name	Name of the party.
votes	Number of votes.
votes_percentage	Percentage of votes in this "okrsok".
preferential_votes	Number of preferential candidate votes.
preferential_votes_percentage	Percentage of preferential candidate votes in this "okrsok".

Table 3. Variables available regarding election results.

2.2 Online data

We are using two different online data sets in this project - a table denoting the names of the parties in the currently ongoing election and a table denoting the heuristic parameters of the vote distribution depending on the current counted percentage[2]. These two respective datasets are located in the

[2]These were obtained from a Monte Carlo simulation where thousands of election nights were sampled. In each sample, the 2020 election vote counts per okrsok were multiplied by a uniformly random variable s, where $s \sim U[a, b]$. Multiple simulations were conducted with different values of a and b, ranging from $a = 0.5$ to $a = 1$ and from $b = 1$ to $b = 1.5$ (uniform priors). The resulting vote distributions' mean, variance and quantiles were then saved for increments of 1% in the counted percentage of votes, resulting in 400 parameters

google sheets **election_data.Quantiles** and **election_data.PartyNames**. Their respective variables are depicted in tables 4 and 5.

counted_percentage	Percentage of counted votes.
lower_quantile	Lower 95% quantile of the vote distribution.
upper_quantile	Upper 95% quantile of the vote distribution.
mu	Expected value of the vote distribution.
sigma_squared	Variance of the vote distribution.

Table 4. Vote distribution parameters obtained from a Monte Carlo simulation.

ID (not in header)	ID of the party.
name (not in header)	Name of the party.

Table 5. Party names dataset.

3 Google sheets structure

We are using two different google sheets in this project. The main sheet is called **election_data** and contains some of the data described in the previous section, a user dashboard v 1.0 with detailed graphs depicting estimates' evolution in time and automatically filled sheets used to keep past estimates and temporary data saved online during the elections night. This sheet is meant mostly for temp data storage. In addition, the graphical depictions of past estimates can be used for a better perspective on prediction reliability. If the curve for a particular party has remained flat for some time now, there is a good chance that this is already quite close to the final estimate.[3] The secondary sheet is called **elections_lite** and it is meant for a quick overview of the current state of the election. It does not contain any graphical elements

(4 parameters in 100 counted vote percentage increments).

[3]This is a very informal way to view the estimate of course. One should primarily take into account the expected vote count and the upper/lower quantiles of the distribution.

which can slow a browser down considerably, instead it contains probabilistic estimates as well as fair computed betting odds for various events. This is the sheet to be used for betting during the actual elections night. Both sheets contain differently colored cells. A red cell corresponds to a model output, a green cell to a model input (data), while a yellow cell corresponds to an intermediate computation within the google sheet itself - usually for the sake of formatting or transposition between the different sub-sheets. Further info on both sheets can be found in Table 6 and Table 7.

4 Code

We use python as our main programming language. The relevant files and functions as well as their description is shown in Table 8. There are two algorithms running in parallel.

The first one - download_data.py - periodically checks for new data. If it finds some, it downloads it and stores within /data/current_data.csv.

The second algorithm - main.py - keeps importing the /data/current_data.csv dataset. If it notices that there are new observations as opposed to the ones stored within election_data.GranularData, it runs the main prediction model, extracts the results, and updates both google sheets via google sheet api.

We shall now look into both algorithms more closely.

4.1 download_data.py

This algorithm takes care of downloading the latest data from the web-page of the Slovak statistics office located at[4]

`https://volby.statistics.sk/nrsr/nrsr2020/`

The algorithm is described in Algorithm 1. In short, it first sets vote counts for each okrsok equal to zero. Afterwards, it downloads aggregated vote

[4]The web address might be different in the next elections.

counts on kraj level. While there is an okrsok with zero votes, it keeps downloading aggregate vote counts first on the level of kraj, then obvod, okres, etc., until it finds the relevant okrsok that has been added to the dataset. Upon finding it, it downloads its granular data and updates the /data/current_data.csv file with it. The end result is a perpetually updated, locally-stored file with the relevant current vote counts for each available okrsok.

Algorithm 1 download_data.py

```
Import historical data.
Assign 0 votes to each okrsok.
Import party names.
while ∃ okrsok with 0 votes do:
    Import latest locally saved vote counts.
    Download vote count by kraj.
    Determine which kraj has new votes.
    if new votes in any kraj then:
        for ∀ kraj with new votes do
            for ∀ obvod with new votes do
                for ∀ okres with new votes do
                    for ∀ town with new votes do
                        for ∀ okrsok with new votes do
                            Add this okrsok values into the locally saved data.
```

4.2 main.py

This algorithm takes care of importing the latest locally-saved dataset, checking whether there are any new observations and if so, running the whole prediction algorithm and uploading the results into both google sheets. The algorithm is described in Algorithm 2.

Algorithm 2 main.py

```
Clear values in election_data google sheet.
Import data from last elections.
Compute aggregate past vote counts by kraj, obvod, okres, town and okrsok.
Compute the adjacency matrix of Slovak towns.
Set the party IDs and names.
Download distributional expectations from election_data.Quantiles.
finished:=FALSE
while finished==FALSE do:
    Import the current data from /data/current_data.csv.
    Import the last iteration's data from election_data.GranularData.
    if new observations in current data then:
        Update attendance for all granularities in google sheets.
        Update election_data.GranularData.
        Compute this year's expected attendance by town.
        Compute this year's expected votes per party.
        Compute the relevant mu, sigma_squared and quantiles.
        ∀ parties A and B, compute P(votes A > votes B).
        For each party, compute the probability of winning.
        Update both google sheets with the new prediction.
        if All towns have 100% attendance then
            finished = TRUE
    else:
        Wait for 3 seconds.
```

5 The Model

Assume we have a set $\mathcal{P} = \{1, 2, \ldots, K\}$ of K political parties and a set $\mathcal{C} = \{1, 2, \ldots, N\}$ of N towns (or cities). Denote by $t \in [0, 1]$ the fraction of precincts (okrsok) for which results have already been submitted. We shall understand t as a temporal variable in our analysis even though we observe it at discrete increments and it is not actually measured in seconds. Despite that, it is a version of information filtration at consecutive steps, therefore a temporal variable. For the purpose of our model, let:

- y_{ijt}, where $i \in C, j \in P, t \in [0, 1]$ be the expected percentage of votes for party j in town i, estimated at time t.
- $\bar{v}_{it}$, where $i \in C, t \in [0, 1]$ be the overall number of counted votes in town i at time t.
- $\hat{v}_i$, where $i \in C$ be the overall number of counted votes in town i in the last election.
- $\hat{c}_{it}$, where $i \in C, t \in [0, 1]$ be the percentage of counted precincts (okrsok) in town i at time t.
- v_{it}, where $i \in C, t \in [0, 1]$ be the expected number of votes in town i after all votes are counted, estimated at time t.
- $\bar{y}_{ijt}$, where $i \in C, j \in P, t \in [0, 1]$ be the percentage of votes for party j in town i counted at time t.

Sheet name	I/O/C	Description
DashBoard	O/C	The main sheet. It contains the user dashboard, showing the current prediction of results. It also shows line charts of previous predictions, such that the evolution of prediction values can be judged by the user.
Quantiles	I	Contains the input data regarding vote distribution, see the Data section for more info.
GranularData	O	Saves the granular data used by the prediction algorithm.
Prediction	O	Stores past predictions with timestamps and counted vote percentages.
LowerPredictionQuantile	O	Stores the past lower prediction quantiles with relevant timestamps and counted vote percentages.
UpperPredictionQuantile	O	Stores the past upper prediction quantiles with relevant timestamps and counted vote percentages.
CurrentResults	O	Stores the current state of the election, no prediction, essentially just the status reported by the media at a particular moment in time.
PartyNames	I	Contains the input data regarding the party names, see the Data section for more info.
UcastKraj	O	Counted votes by kraj
UcastObvod	O	Counted votes by obvod
UcastOkres	O	Counted votes by okres
UcastTown	O	Counted votes by town
UcastOkrsok	O	Counted votes by okrsok
LastPrediction	O	Stores lagged predictions with timestamps and counted vote percentages.

Table 6. Characteristics of different sub-sheets for the election_data google sheet. Contains the sub-sheet name, its function (Input, Output, Computation) and its general description.

Sheet name	I/O/C	Description
DashBoard	O/C	The main sheet. It contains the user dashboard, showing the current prediction of results. It also shows P(win) for each party and the associated fair betting odds computed as 1/P(win).
FairOddsComparison	O	Shows fair odds for events of type "party A votes > party B votes".
ProbabilityComparison	O	Shows probabilities for events of type "party A votes > party B votes".
UcastKraj	O	Counted votes by kraj
UcastObvod	O	Counted votes by obvod
UcastOkres	O	Counted votes by okres
UcastTown	O	Counted votes by town
UcastOkrsok	O	Counted votes by okrsok
PartyNames	O	Contains party names, used in elections_lite.DashBoard, elections_lite.FairOddsComparison and elections_lite.ProbabilityComparison to fill in the party names.

Table 7. Characteristics of different sub-sheets for the elections_lite google sheet. Contains the sub-sheet name, its function (Input, Output, Computation) and its general description.

File	Description
config.py	Stores variables and settings.
download_data.py	Downloads new data.
formatting_functions.py	Various auxiliary formatting functions.
google_api_functions.py	Tools for google sheet management. Append, read, write, delete.
location_functions.py	Tools used with location data on Slovak towns.
main.py	Checks if new data is available, if so, updates the prediction.

Table 8. Characteristics of different sub-sheets for the elections_lite google sheet. Contains the sub-sheet name, its function (Input, Output, Computation) and its general description.