

Introduction

Ignacio Amaya and Justin Vailhere

Introduction

In Spain the national elections take place every four years. It is important to analyse the results each political party has obtained.

Spain is divided into 17 autonomous communities and two autonomous cities. Each of those communities is formed by one or several provinces. In total, Spain has 50 provinces. Each of these provinces and communities yields different election results that are important to understand how the votes are spread across the Spanish population.

Merging demographic and political data is also an option. This can help us to gain very useful knowledge about how different demographic factors influence in the rise or fall of a specific political party.

Objectives

- Get summarized information about the results of the Spanish elections in different years.
- Compare the distribution of the votes for the different political parties.
- Look up figures about the results.
- Get detailed information about the different autonomous communities and provinces.
- Check the evolution of a party through all the elections.
- Compare the evolution of a party with different demographic factors.

Dataset

We've used datasets from the INE (<http://www.ine.es/>).

The main problem has been that the data wasn't ready to be analysed and we've lost a huge amount of time cleaning and merging the datasets.

Another important issue has been that the codes of the different provinces and communities didn't match with the maps we decided to use. Those maps were obtained using the GDAL library.

Techniques used

In order to decide which techniques to use we've taken into consideration three questions. These questions are very important if we want to achieve our objectives:

- WHAT do we want to know from our plots?
- HOW are going to present and encode our data?
- WHY our users would use our app?

All that questions are closely related with the objectives we previously set.

We've used a top-down approach in our case. This means that we developed the app thinking in what people would like to know about the Spanish elections and then we picked the techniques we considered the best ones to accomplish those goals.

WHAT: Data abstractions

We've used positions and attributes. When dealing with spatial data it is important to place that information into space, because position is the most important visual channel. That's why when presenting the results of elections giving only the figures is not enough. You need to plot the map and associate each province with its winning party, so that way the user can detect in a glance the outcome of the elections and how the votes are distributed.

Our datasets were simple and had categorical attributes, such as the names of the parties or the names of the regions, but also numerical attributes, such as the percentage of votes. The main problem with this was to keep track of the regions, which weren't consistent in our different datasets. Also, the names of the parties needed some cleaning and it was necessary to pre-process the data to show only the most important results. The small parties weren't important so they were gathered together to be presented tagged as others.

The library we used for the geographical data was based on polygons and was very difficult to deal with because it wasn't easy to merge with our other data. Presenting Spain together was important, so the data had to be modified to allow a united representation of Spain (including the Canary Islands and the autonomous cities).

Fortunately, the demographic data was consistent with the elections result, so its pre-processing wasn't so hard.

WHY: Task abstractions

- Actions: we can divide the actions of our application in three different categories.
 - **Use:** Users of our app can discover new information about how the demographic factors of the provinces could be related with their results. However, our app can't produce new information. It would be a nice extra feature to implement. Machine learning could be apply to maybe predict the results of the incoming elections knowing the demographic data.
 - **Search:** Users can look up the figures of the elections and browse to find the winners in each province.
 - **Query:** Users can compare different demographic factors and select the province they are interested in.
- Targets: the users can find trends in data and check the distribution of the parties at a national level or in each region.

HOW: Design choices

We decided to use three different plots to achieve our predefined objectives.

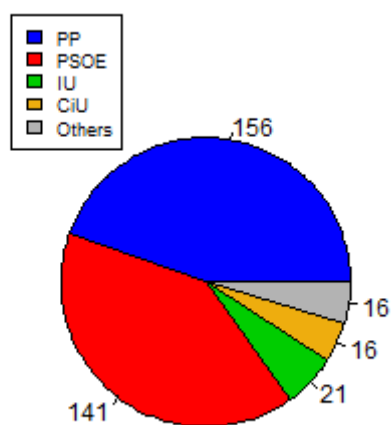
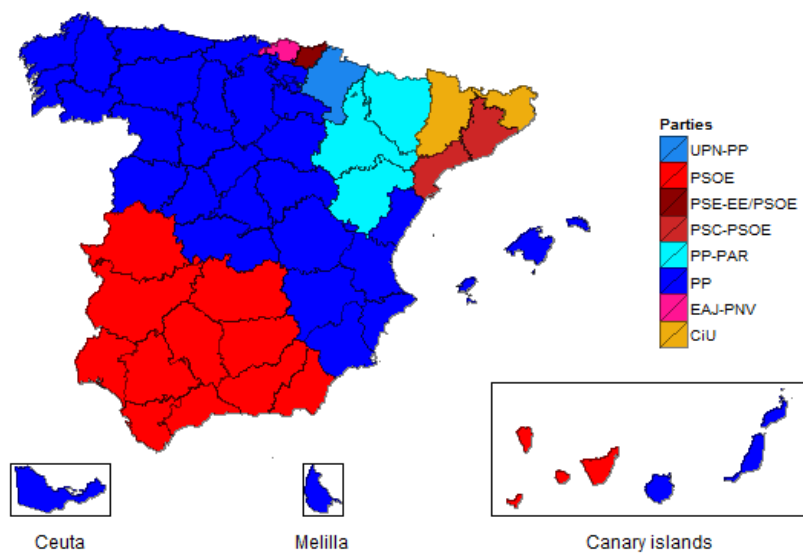
- Regions map: to represent the winning parties in each community.
- Pie chart: to represent the distribution of seats per community.
- Grouped bar chart: to show the percentage of features in each province.
- Stacked line chart: to show how the parties evolved through the years and compare them with the evolution of the demographic features in the same period.

Tool

We've divided the app in two views. The first one will show information about one year, while the other will show the evolution throughout all the years.

View 1

The main objective when designing this view was being able to perceive the map all the time. That's why it is a key part in the design of this view. We can see the details about the elections below the map summarized in a table. On the right we have two tabs: the first one to see the distribution of the results in a pie chart and the other to check how the different demographic features are distributed in the provinces. In the next page we can see an example of this elements.



	partyName	number	seats	percentage
696	PP	9716006	156	38.8
697	PSOE	9425678	141	37.6
698	IU	2639774	21	10.5
699	CiU	1151633	16	4.6
709	Others	369404	16	1.5

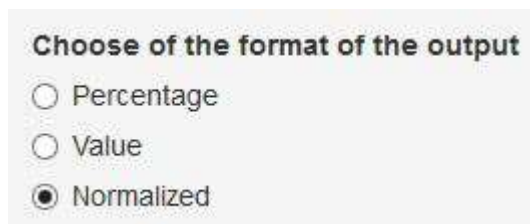
Showing 1 to 5 of 5 entries

View 2

This view allows the user to easily manipulate the evolution of the results of the party in the county he wants to see if there is any trending and to compare it with some general information to see if there is any correlation.

The window is divided in two parts: the graph and a set of options to manage the data the user wants to see.

First he can choose the format of the output: the percentage, the value and the normalized data. The Percentage and Normalized option will sometimes plot nothing because some values are missing in the data set.



Choose of the format of the output

☐ Percentage

☐ Value

☒ Normalized

Secondly, he can choose the county where he wants the results and the features to compare.



Choose a party to display

Partido Popular (PP) ▼

Then, he chooses the party. He can also take the general result (participation, number of null votes...).



Feature(s) to compare

☐ Drug dependencie

☐ Alcohol dependencie

☐ Constitutive pensions

☒ Prisonners

☐ No educated

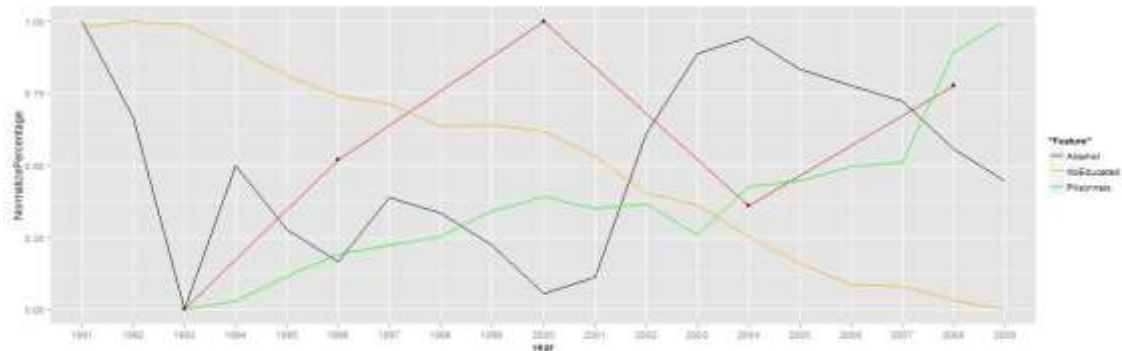
☐ Educated

Finally, we have the features we want to compare. He can choose multiple features to compare with the results. But this option is only available with the normalized output.

The graph

The graph is a scatter plot matrix: this allows the user to find trend and correlations easily between up to 12 attributes and hundreds of items. And to facilitate the task we put all the plots on the same graph.

The results of the election will be in red.



Limits

The data limits us. Indeed we have only 5 elections results usable because our data on the features covert only the years 1991 to 2009. We need data from 1970 to 1991 to build a really efficient visualization tool on the elections in Spain and also more variety in our data, such as unemployment, average wages...

Conclusions

We've had a lot of problems with the bad quality of the datasets, so we've learn to pick more carefully those before starting a project.

Also, we haven't finished completely the app due to those problems. That's why some labels are not still there.

In general we've learned that creating a good visualization tool is not so easy. There have been a lot of options taken into account when designing the app and deciding which one to take is a hard decision.