Information Retrieval and Text Mining Report

Text Mining the Origins of Spanish surnames

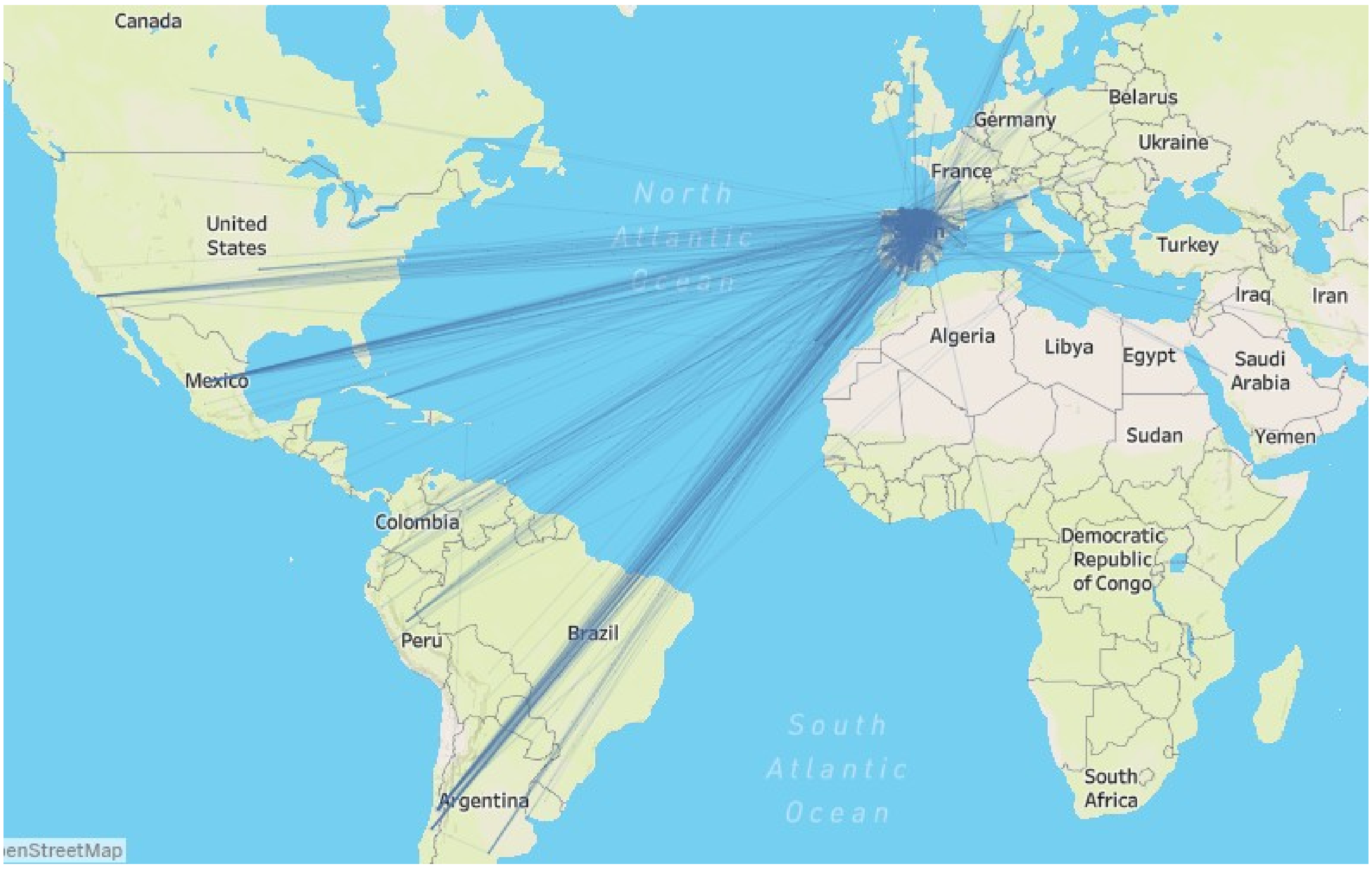
Enrique Barrueco Mikelarena

Department of Knowledge Engineering Faculty

May 27th 2020

Introduction

This project text mines the origins of Spanish surnames aswell as places where each surname expanded to. This task is perform by two classfiers. To build these classifiers this project uses weak supervision (WS) to help alleviate the data bottleneck problem. After data is found about the history of a surname, more information is mined from the DBpedia knowledge graph, of historical entities, in particular, the place of birth and place of birth of entities recogznied as people in spacy that where born before 1920. These places are also included in the place where the origin expanded to, this results are plotted on a world map using Tableau, where several queries can be made, this is simply all the connections form origin to destinations.



Background

My corpus is formed by a collection of texts describing the origin and history of 629 surnames obtained from the website www.quesignificaelapellido.com the average length of these text is 1491 characters.

Entity Recognizion:

Because my focus is to detect the origin of surnames I need to detect the mention of any surname a new entity type that spaCy does pick up by default, to do this I created a new entity type, ‘SURN’, to train this new category as well as to increase the how well spaCy recognized and people. To avoid the catastrophic forgetting problem I mixed in examples of other entity types that spaCy correctly recognized before, and I am correcting the ones it didn't recognize correctly.

The model I am going to use as a base and build on is spaCy’s ***es_core_news_md*** Spanish model, which is a **CNN** trained on the **AnCora and WikiNER corpus.** It worked well by default on places, but after training it recognized complex names of people better, this are the results of how well the ipdated model recognizes surnames:

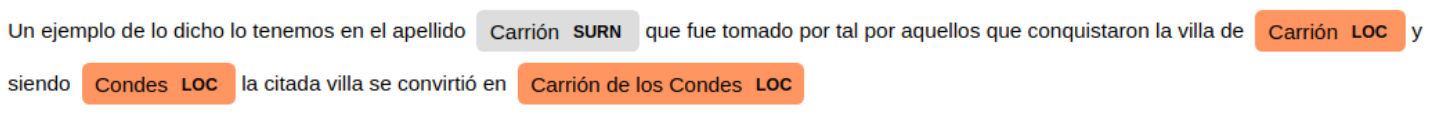
Metrics SURN

	0	1
Precision:	0.941	0.785
Recall:	0.842	0.916
F1:	0.888	0.842
support:	19	12

Confusion matrix

	Actual values	
Predicted Values	No ‘SURN’	'SURN’
No ‘SURN’	16	3
'SURN’	1	11

It looks like the model ignores some surnames but works good enough. An example of how the model is able to correctly differentiate when a word is used as a surname or as a location.


Un ejemplo de lo dicho lo tenemos en el apellido Carrión SURN que fue tomado por tal por aquellos que conquistaron la villa de Carrión LOC y siendo Condes LOC la citada villa se convirtió en Carrión de los Condes LOC

These are the results of testing how well the spaCy model regonizes locations, entity ‘LOC’, on complex sentences:

Metrics LOC

	0	1
Precision:	0.818	0.95
Recall:	0.9	0.904
F1:	0.857	0.926
support:	10	21

Confusion matrix LOC

	Actual values	
Predicted Values	No ‘LOC’	'LOC’
No ‘LOC’	9	1
'Loc’	2	19

After this model has been updated with the training examples provided, this new model is saved to a directory of the local storage, and can be read from disk. This model is included in the attached files.

Pre-processing

Once spaCy was able to correctly recognize the entity ‘SURN’ there was still some pre-processing to do before writing the labeling functions that will build the classifiers.

Because the labeling functions I will build are based on recognizing entities and keywords rather that semantic structures I want to **resolve co-references of the surname** even if they make the sentence slightly grammatically incorrect or awkward.

The type of co-reference resolution that is most neened when the text is talking about the origin of a surname, the author is
To resolve mentions of the surname origins that instead of mentioning the surname mention a word meaning thw word surname a series of conditions are set to catch different gramaticall structures that talk about a surname, in the form of:

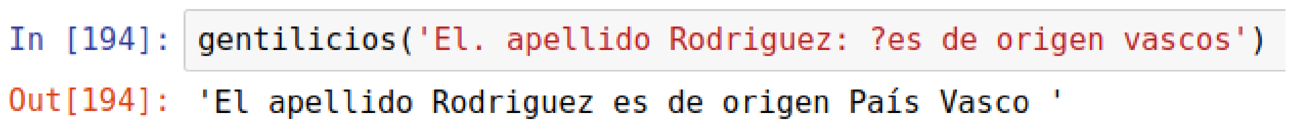
```
if 'el apellido' in sentence.lower() or 'el linaje' in sentence.lower() or 'este
apellido'…
```

To resolve mentions of the surname’s origin that don’t explicitly say that the surname was from X but instead the location is referred to by its demonym (‘gentilicios’ in Spanish), I wrote the function *gentilicios(sentence,surname)* that returns the resolved sentence and a binary variable that indicates if it has been modified or not.

It perform a number of pre-processing operations such as:

→ Strip the word of extra punctuation: To do this I used python’s *translate* instead of creating a REGEX expression because *regular expressions* do not beat the performance of *translate* for this task

→ If a demonym is encountered it is replaced by the location: Every word is lemmatized using spaCy’s native lemmatization tool, and the lemma is checked to be in a dictionary of Spanish demonym which is included in the data submited.



```
In [194]: gentilicios('El. apellido Rodriguez: ?es de origen vascos')
Out[194]: 'El apellido Rodriguez es de origen País Vasco '
```

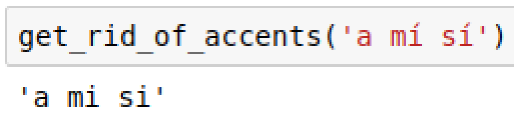
In this example there were a number of mistyped punctuation signs and the mention of ‘***vascos***’, which is the plural form of the demonym of people and things born in the ***Basc Country***, this is correctly corrected to ***País Vasco*** by the function because the word had been lemmatized first to match with the entry in the dictionary.

In this example the surname ‘Rodriguez’ was already present in the sentence, but if it wasn’ t, it would have been resolved during the running of the algorithm. Because whenever there is a mention of the word surname and a SURN entity is not present in the sentence, then the mention of the word surname is replaced by the surname of that the text originated from. This will increase the recall and precision of my labeling functions.

Another way to increase recall is to strip accents (‘) form the sentence, using the function get_rid_of_accents which uses the following regular expression:

```
re.sub(r"([^n\u0300-\u036f]|n(?!\u0303(?![\u0300-\u036f])))[\u0300-\u036f]+", r"\1",
                        normalize( "NFD", sentence), 0, re.I)
```

This is an example of how it works:



```
get_rid_of_accents('a mí sí')
```

```
'a mi si'
```

Processed Corpus

The original corpus had a length of *7643* sentences, a number of those were simply promotional messages for buying merchandise from the website were I scraped the text, this sentences were excluded using matching expressions. In total 1154 of these sentences were dropped, so the size of the corpus after the pre-processing is reduced to *6489* sentences.

Labeling functions:

I created a series of labeling functions with Snorkel that will help me generate labeled data for setting up the classifiers I need to decide if a sentence contains the origin of a surname or not and if it contains the mention of a surname expanding.

Labeling functions (LFs) are heuristics that take as input a data point and either assign a label to it (in this case, 1 for origin of surname or 0 not origin of surname) or abstain (don't assign any label). This Labeling functions can be noisy: they don't have perfect accuracy and don't have to label every data point.

After writing and refining them after seeing how they labeled data this are the LFs I used to build a classier for the origin of surnames:

1) ***has_surname_origin***:

```
This labeling function returns 1 i.i.f  all the following conditions meet:
        Any enity in the sentence is a SURN.
        The text of any of the entities in the sentence is equal to the surname
        associated with that sentence.
        Any entity in the sentence is a ‘LOC ‘ (Location) is not included in banned
        places.
Else:
        Abstain.
```

2) ***mentions_family***:

```
This labeling function returns 1 i.f.f the two following conditions meet:
        Any of : 'familia', 'apellido', 'linaje' or 'linajes' are mentioned in the
        sentence.
        Any of the entities in the sentence is a Location.
Else:
        it returns -1
```

3) ***mentions_root***:

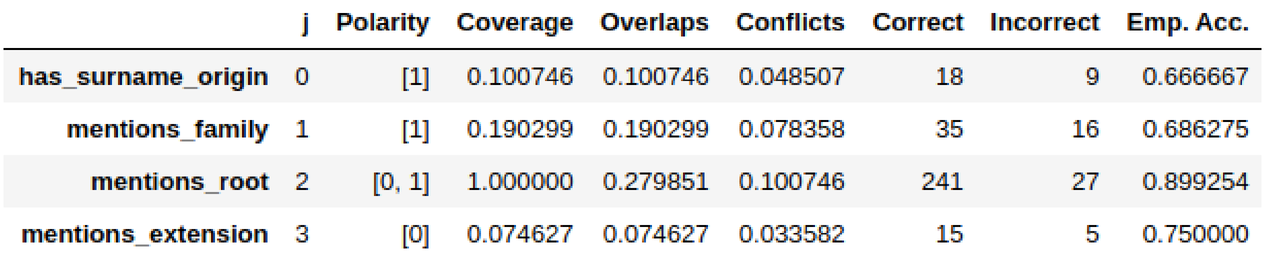
```
This labeling function returns 1 i.f.f the two following conditions meet:
        Any of  the lemmatized words in the sentence are in the list root.
                This words relate to the mention of the origin of a surname.
                Ordinal Anaphoras are taken into account as keywords.

        Any of the entities in the sentence is a Location.

Else:
        it returns Abstain
```

4) ***mention_extension***:

```
        This labeling function returns 0 i.f.f the two following conditions meet:
                Any of  the lemmatized words in the sentence are in the list extension.
                        This words relate to the mention of the surname expanding to other
                        countries and cities within Spain of a surname.

                Any of the entities in the sentence is a Location.

        Else:
                it returns Abstain
```

It is important to keep in mind that with these I am prioritizing high precision over recall, with the hope that the classifier will pick up more patterns and increase recall, and knowing that the text of each surname contains, for the most part, many sentences.

Summary statistics of the labeling functions



	j	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	Emp. Acc.
has_surname_origin	0	[1]	0.100746	0.100746	0.048507	18	9	0.666667
mentions_family	1	[1]	0.190299	0.190299	0.078358	35	16	0.686275
mentions_root	2	[0, 1]	1.000000	0.279851	0.100746	241	27	0.899254
mentions_extension	3	[0]	0.074627	0.074627	0.033582	15	5	0.750000

Majority Model

It is possible to build a model based on the output of these 4 labeling functions, the simplest one is the majority model, which simply classifies based on the majority vote among the labeling functions. This model was tested on labeled data used for testing and had the following metrics:

f1 score: 0.523; **roc-auc**: 0.871; **precision** score: 0.363; **recall**: 0.933

This model has very high recall by very low precision because it counts every vote with an equal weight, however, as can be seen by looking at the summary statistics of the LFs on the last image, they have varying properties and should not be treated identically. In addition to having ranging accuracies and coverages, some of them are correlated, resulting in certain signals being over-represented by this model. To handle these issues appropriately Snorkel provides a more sophisticated LabelModel.

Label Model

The goal is now to convert the labels from the LFs into a single *noise-aware* probabilistic (or confidence-weighted) label per data point, which are re-weighted combinations of our labeling function's votes different from what a majority model would predict.

This are the metrics of that Label Model:

Metrics Label Model

	0	1
Precision:	0.92	0.81
Recall:	0.95	0.72
F1:	0.94	0.76
support:	208	60

Confusion matrix Label Model

	Actual values	
Predicted Values	0	1
0	198	10
1	17	43

The model seem to be pretty good at detecting the origin of surnames, it looks like the automatic reweighing of the Lfs increased precision while not lowering recall that much.

Now I will use the outputs of the LabelModel as training labels to train a classifier which can generalize beyond the labeling function outputs to see if we can improve performance further. I wil use the probabilistic training labels we generated in the last section to train a classifier for our task. That is, the probabilies of a sentence being calssified as 0 or 1. This output can be used with much more complex, training data-hungry models. However, because of my time to label and obtain data as well as computational power is limited I will restrict my model selection to computationally simple models.

Linear Models:

I will use a **Logistic Regression** as a baseline:

Because I am working with text before fitting a logistic model to my data I need to convert text into a matrix of token counts, to do this I will use **CountVectorizer** and then fit transform from sklearn. After the text has been vectorized by token count a Logistic Regression is fit and predicts on the test matrix. This are the results:

Metrics Logistic Model

	0	1
Precision:	0.89	0.86
Recall:	0.97	0.6
F1:	0.93	0.71
support:	208	60

Confusion matrix Logistic Model

	Actual values	
Predicted Values	0	1
0	202	6
1	24	36

Vectoring using TfidfVectorizer produces very similar slightly less optimal results.

This are the **top words by count frequency excluding stopwords:**

don	linaje	apellido	rey	origen	casa	real	oro	año	caballero
48	33	32	20	20	18	14	14	14	13

Using generated labels to Get features from BERT

Since I have very limited training data even after generating labels with Snorkl, I cannot train a LSTM with a lot of parameters. Instead, I will use a pre-trained BERT model to generate embeddings for each my sentences, and treat the embedding values as features.

Now, I train the simple logistic regression model I trained before but this time using the BERT features, obtained using labels from my Label Model instead of using directly the probabilities of the Label Model.The results are not better than previous models.

Metrics Logistic Model BERT features

	0	1
Precision:	0.89	0.91
Recall:	0.91	0.6
F1:	0.9	0.63
support:	208	60

Confusion matrix Logistic Model BERT features

	Actual values	
Predicted Values	0	1
0	189	19
1	24	36

A **Naive Bayes Model** was also tested, it had the following sub-optimal metrics:

```
f1 score: 0.645; precision: 0.5; recall: 0.909
```

Lastly, a **Support Vector Machine** model was fit to the Tfidf Vectorized text, the SVM had the following metrics:

Metrics SVM

	0	1
Precision:	0.82	1
Recall:	1	0.12
F1:	0.9	0.22
support:	65	16

Confusion matrix SVC

	Actual values	
Predicted Values	0	1
0	65	0
1	14	2

As we can see this model is very conservative and outputs alsmot only 0s.

Detecting expansion of a surname

After trying all the different models, it turns out that the best performing one is the origin Label Model built from the labeling functions, it is the one that will be use to predict whether the sentences of my corpus contain the origin of the surname.

No I repeat the hole process to build another classfier, this one classfies sentences as talking about the extension of the surname to other places or not. For this I built two labeling functions:

1) ***mentions_family***:

```
This labeling function returns 1 i.f.f the two following conditions meet:
        Any of  the lemmatized words in the sentence are in the list familias,
        which refers to mention of families and lineajes.

        Any of the entities in the sentence is a Location.

Elif there is a LOC entity:
        it returns Abstain
Else:
        it return 0
```

2) ***mention_extension***:

```
This labeling function returns 0 i.f.f the two following conditions meet:
        Any of  the lemmatized words in the sentence are in the list extension.
                This words relate to the mention of the surname expanding to other
                countries and cities within Spain of a surname.

        Any of the entities in the sentence is a Location.

Elif there is a LOC entity:
        it returns Abstain
Else:
        it return 0
```

I use the **Majority Model** to make predictions. Because both labeling functions can output any of 3 possibilities a Majority Model does well. The goal of this classifier is to have high recall, precision is not as important as when finding the origin of a surname.

Once both classifiers are selected a **Greedy algorithm loops trough the corpus** looking at the labels that were assigned to the sentences. For the set of sentences corresponding to the text of a surname **only one origin can exist**, however, more than one of the sentences might have been labeled as being the origin of the surname.

Logically, by looking at the structure of this text, **the origin of the surname is often one of the first things mentioned**, and therefore among the first sentences of that text. Because of this the algorithm greedily looks to assign the origin of the surname to **the first location mentioned** on the first **sentence labeled as containing the origin of the surname** if this location is not in a list of banned places. After that, any other place mentioned in any sentence labeled as either talking about the extension of the surname or containing the origin of the surname is appended to a set of other places where the surname expanded. To increase the information extracted out of these key sentences I will also extract information from DBpedia using SPARQL.

Entity linking:

To extract additional structured information from those sentences I need to link the entities recognized by my model to entities on Wikipedia.

The library I will use to implement NEL is **DBpedia Spotlight**. Target knowledge base for NEL here is DBpedia. After an entity has been linked to its URI on DBpedia where I can potentially extract additional information about that entity that can be useful to understanding how surnames spread.

When the algorithm encounters a sentence that has been labeled as a 1, it loops throuh the entities in that sentence, if any of them is of type person, ’PER’, a query is made to wikidata using the pipeline function *info_dbpedia* to try to extract the entities place of birth and place of death, it also queries the date of birth and death, if an entity was born after 1920 it is not considered hitorically relevant and is skipped, otherwise, the algith includes the place of birth and place of death of that entity, and those places are included in the places where the surname expanded.

Therefore, when a sentence is passed to the *info_dbpedia* a list of locations is returned. This is an example with the sentence:

“Estando primeramente en **Cuba** con su pariente **Diego Velázquez**, participó en la primera expedición a **Méjico** con **Francisco Hernandez de Córdoba**; en la segunda, con **Juan de Grijalva** y en la tercera, que fue la definitiva, junto a **Hernán Cortés**.”

These are all the entities Spotlight recognized.

```
{'Cuba': 'http://es.dbpedia.org/resource/Cuba',
'Diego Velázquez': 'http://es.dbpedia.org/resource/Diego_Velázquez',
'Méjico': 'http://es.dbpedia.org/resource/México',
'Juan de Grijalva': 'http://es.dbpedia.org/resource/Juan_de_Grijalva',
'Hernán Cortés': 'http://es.dbpedia.org/resource/Hernán_Cortés'}
```

This are the results of only querying the entities of type person using queries of the form:

```
‘’’
PREFIX dbo: <http://dbpedia.org/ontology/>

select * where {

                %s
                }
```

```
'''
%queries[i][g]

with:

queries[i][g] = <http://es.dbpedia.org/resource/Juan_de_Grijalva> dbo:deathDate ?deathDate.
```

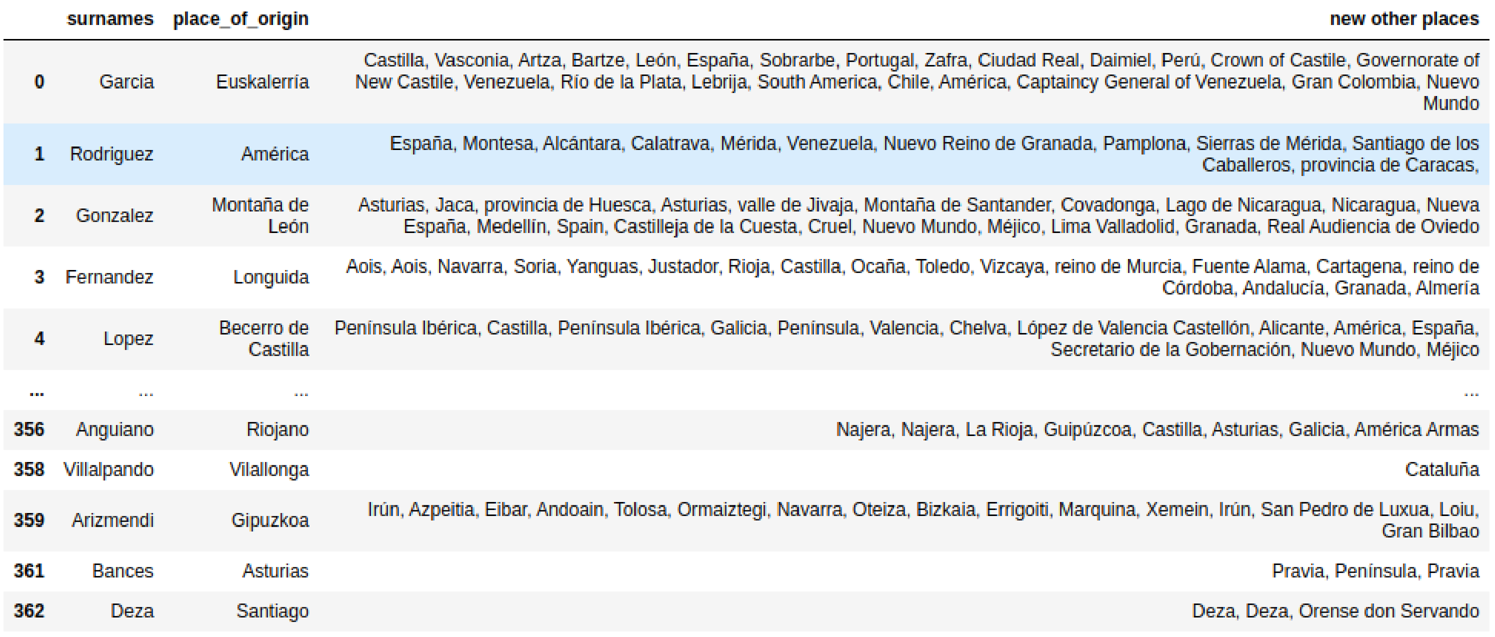
The results are:

```
Juan_de_Grijalva:
{'birthPlace': 'http://dbpedia.org/resource/Cuéllar',
'deathPlace': 'http://dbpedia.org/resource/Nicaragua',
'deathDate': '1527-1-1',
'birthDate': '1490-1-1'}

Diego Velázquez:
{'birthPlace': 'http://dbpedia.org/resource/Seville', '
deathPlace': 'http://dbpedia.org/resource/Madrid',
'deathDate': '1660-08-06',
'birthDate': '1599-06-06'}

Hernán Cortés:
{'birthPlace': 'http://dbpedia.org/resource/Medellín,_Spain',
 'deathPlace': 'http://dbpedia.org/resource/Castilleja_de_la_Cuesta',
'deathDate': '1547-12-02',
'birthDate': '1485-1-1'}

In this case all the entities where historically relevant and the place where
they were borned and where they died does add information to my knowledge base.
```

The output of the function *info_debepedia* for this example *is:*

```
['Seville', 'Madrid', 'Cuéllar', 'Nicaragua', 'Medellín, Spain', 'Castilleja de la
Cuesta']
```

This is a look of how the data looks after the algorithm is done:



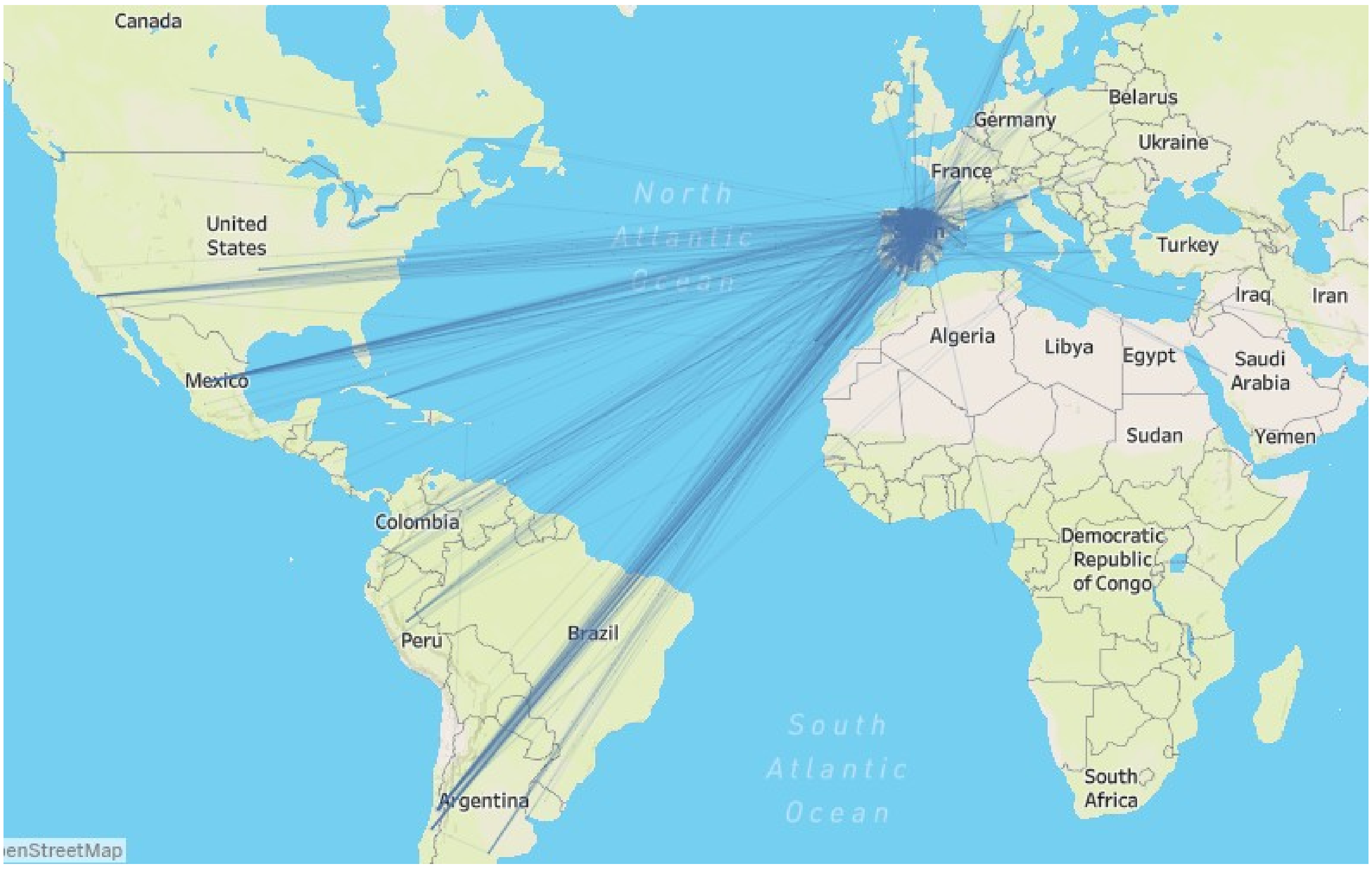
	surnames	place_of_origin	new other places
0	Garcia	Euskalerría	Castilla, Vasconia, Artza, Bartze, León, España, Sobrarbe, Portugal, Zafra, Ciudad Real, Daimiel, Perú, Crown of Castile, Governorate of New Castile, Venezuela, Río de la Plata, Lebrija, South America, Chile, América, Captaincy General of Venezuela, Gran Colombia, Nuevo Mundo
1	Rodriguez	América	España, Montesa, Alcántara, Calatrava, Mérida, Venezuela, Nuevo Reino de Granada, Pamplona, Sierras de Mérida, Santiago de los Caballeros, provincia de Caracas,
2	Gonzalez	Montaña de León	Asturias, Jaca, provincia de Huesca, Asturias, valle de Jivaja, Montaña de Santander, Covadonga, Lago de Nicaragua, Nicaragua, Nueva España, Medellín, Spain, Castilleja de la Cuesta, Cruel, Nuevo Mundo, Méjico, Lima Valladolid, Granada, Real Audiencia de Oviedo
3	Fernandez	Longuida	Aois, Aois, Navarra, Soria, Yanguas, Justador, Rioja, Castilla, Ocaña, Toledo, Vizcaya, reino de Murcia, Fuente Alama, Cartagena, reino de Córdoba, Andalucía, Granada, Almería
4	Lopez	Becerro de Castilla	Península Ibérica, Castilla, Península Ibérica, Galicia, Península, Valencia, Chelva, López de Valencia Castellón, Alicante, América, España, Secretario de la Gobernación, Nuevo Mundo, Méjico
...	...	...	...
356	Anguiano	Riojano	Najera, Najera, La Rioja, Guipúzcoa, Castilla, Asturias, Galicia, América Armas
358	Villalpando	Vilallonga	Cataluña
359	Arizmendi	Gipuzkoa	Irún, Azpeitia, Eibar, Andoain, Tolosa, Ormaiztegi, Navarra, Oteiza, Bizkaia, Errigoiti, Marquina, Xemein, Irún, San Pedro de Luxua, Loiu, Gran Bilbao
361	Bances	Asturias	Pravia, Península, Pravia
362	Deza	Santiago	Deza, Deza, Orense don Servando

Out of the 629 texts describing surnames, an origin was found for 321 of them, with every origin having an average of 15 destinations associated with them.

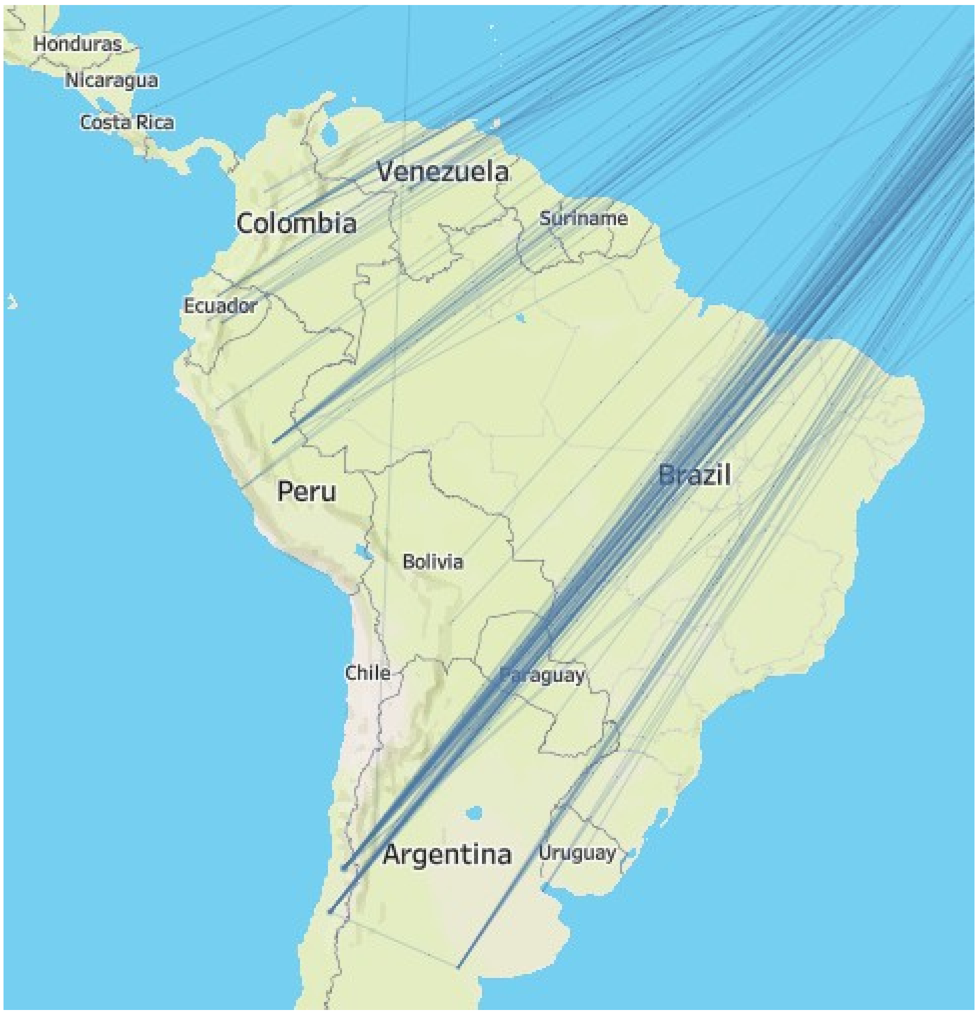
In order to plot this data I needed to do some pre-precessing to have the data look like the next figure, the code using for this is quite intuitive and is included on the jupyter notebook, which is the format the Tableau recognizes the easiest and is then easiest to plot the paths. To obtain the latitude and longitude of each of the locations I used GoogleMaps Geocoding API.

	Surnames	Origin_or_Destination	Location	Location_Name	Path_ID	latitude	longitude
2	Garcia	Origin	Euskalerría	Euskalerría_0	Euskalerría_0-Vasconia_2	42.814	-1.66586
3	Garcia	Destination	Vasconia	Vasconia_2	Euskalerría_0-Vasconia_2	43.3143	-1.98342
4	Garcia	Origin	Euskalerría	Euskalerría_0	Euskalerría_0-Artza_3	42.814	-1.66586
5	Garcia	Destination	Artza	Artza_3	Euskalerría_0-Artza_3	43.4181	-2.72402
6	Garcia	Origin	Euskalerría	Euskalerría_0	Euskalerría_0-Bartze_4	42.814	-1.66586
...	...	...	...	...	...	...	...
4669	Deza	Destination	Deza	_Deza_1	Santiago_320- Deza_1		
4670	Deza	Origin	Santiago	Santiago_320	Santiago_320-Deza_2	-33.4489	-70.6693
4671	Deza	Destination	Deza	Deza_2	Santiago_320-Deza_2	37.8804	-4.79819
4672	Deza	Origin	Santiago	Santiago_320	Santiago_320-Orense don Servando_3	-33.4489	-70.6693
4673	Deza	Destination	Orense don Servando	Orense_don_Servando_3	Santiago_320-Orense don Servando_3	42.3358	-7.86388

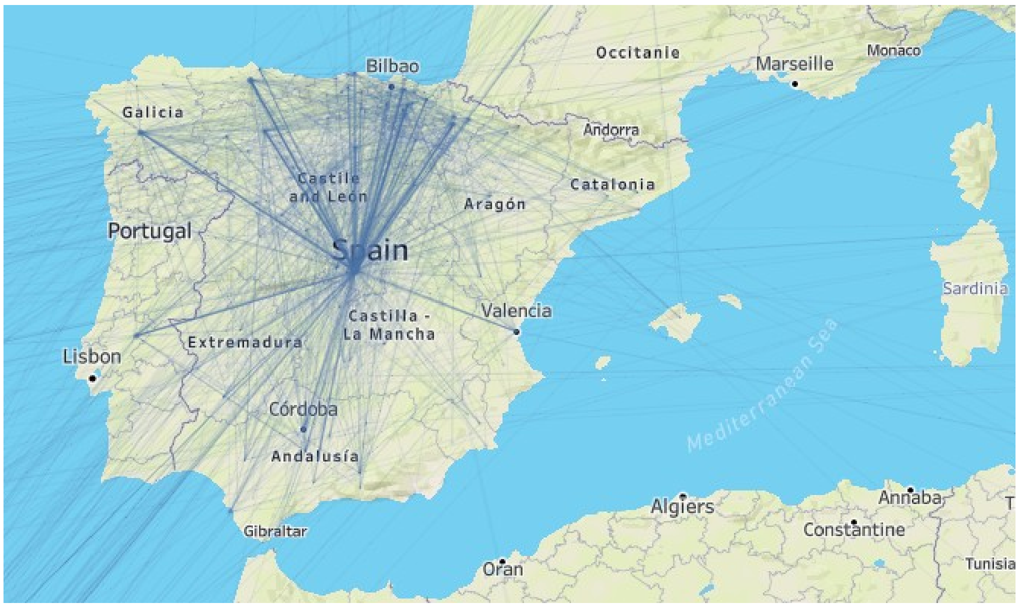
The data can now be finally be plotted using Tableau, in this next image I am showing all the surnames, from its origin to all its destinations. I faded the opacity enough so that the places that are highlighted the most are the ones with the most overlap of connections.



This map is full of interesting insight that I will continue to work on, taking a more detailed look at Latin America one can see the places where the most Spanish people emigrated to:



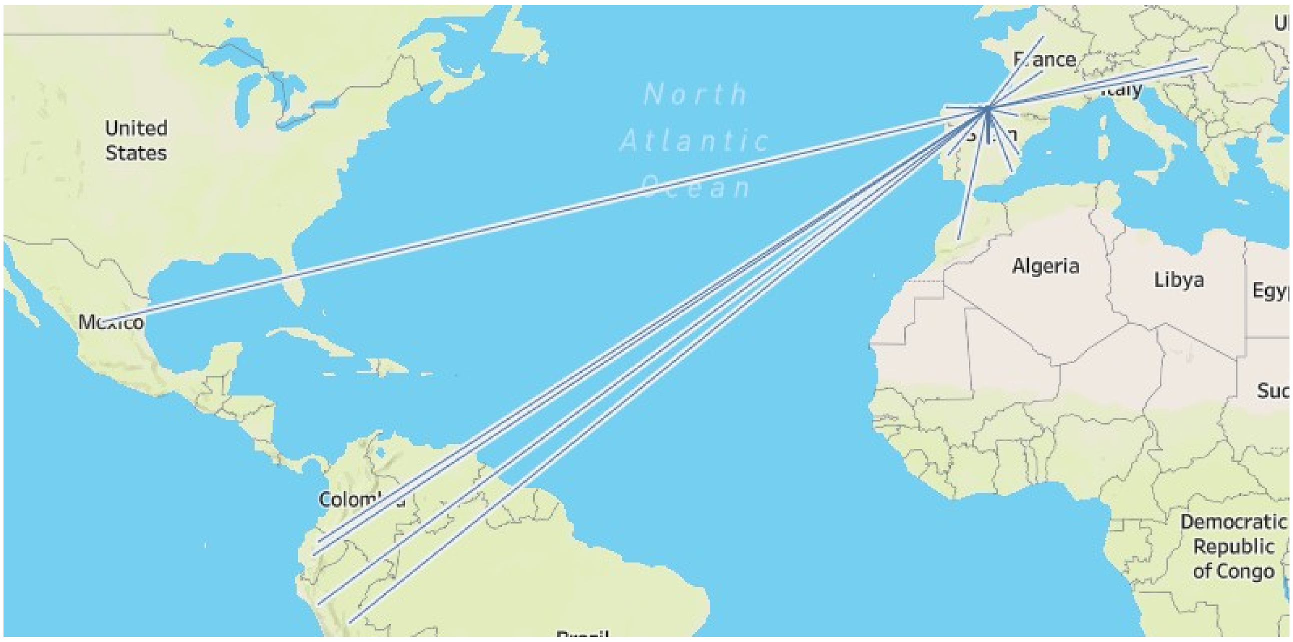
Chile and Argentina have the highest number of connections, which are quite concentrated in the main cities, while the connections in Venezuela, Ecuador or Colombia are more spare. It is also interesting to look at the spider web like map of the extension of surnames within Spain:



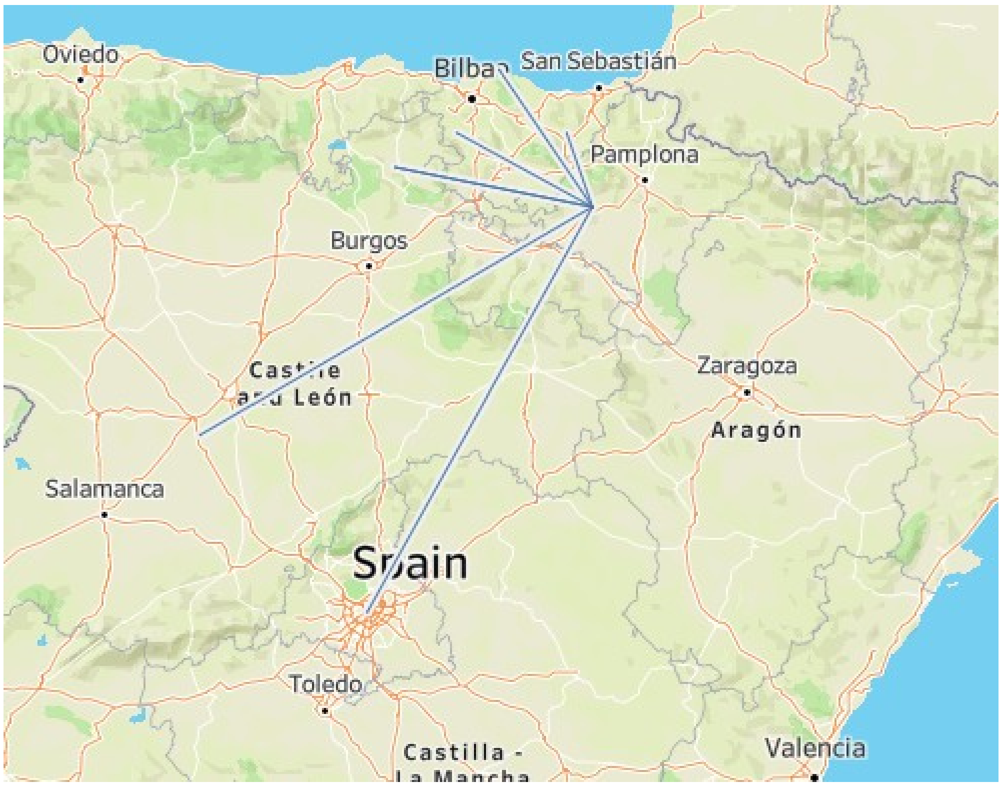
This map contains information that reveals a lot about the social and economic structure within Spain and Latin America. I wanted to also include the date of each of the movements to the map as a value on each edge, but I have not have enough time to develop it. Is something I will continue to work on.

The resulting map is interactive, and one can filter by variables such as the Surname or the Location. Here are a few more examples of interesting insights by filtering by a particular surname:

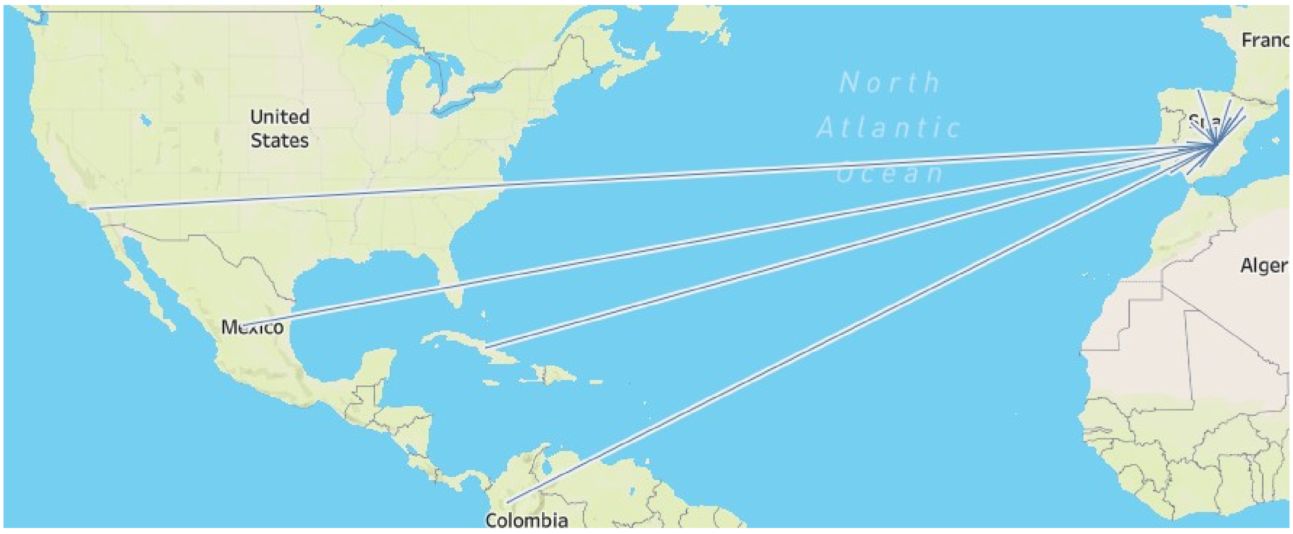
Surname: **Arce**



Surname: **Arellano**



Surname: Serrano


United States
Mexico
Colombia
North Atlantic Ocean
Franc
Spa
Alger