First Part: COVID-19 in Spain

Advanced Regression And Prediction

Roberto Jesús Alcaraz Molina

09/05/2021

Contents

1 INTRODUCTION

The aim of this project is to analyze and try to predict the confirmed cases and deaths due to the coronavirus pandemic (COVID-19) in Spain using regression tools. It started the 31st of January of 2020 in La Gomera (Canary Island) and continues until now, having almost 2.5 million confirmed cases and around 67k deaths.

All the analysis of this project, from the data cleaning until the model results will be done mainly with the `tidyverse` (Wickham et al. 2019) and `tidymodels` (Kuhn and Wickham 2020) packages. The first one is well known for `R` users but the second, even though is still in development, has enough tools for modeling and machine learning.

In the second section, we will see from where we obtained the data and how to clean it. In the third one, we will explore our data by doing some visualizations and we will think about how to adress the problem. In the fourth one, we will do some feature engineering, i.e., we will modify and create some variables, divide our training data into folds to compare our models, and many other necessary steps before start modeling. In the next one, we will tune our model parameters and fit the models. Finally, in the last section, we will select the best model and say some conclusions.

2 DATA CLEANING

The data has been taken from the R package `COVID19` (Guidotti and Ardia 2020). This COVID-19 data set contain 446 observations, which are the days from the 22nd of January 2020 until the 11th of March 2021, with 36 variables divided in 5 categories: *identifiers*, *COVID-19 variables*, *policy measures*, *geographic information* and *external keys*.

Regarding the identifiers, we will select only date. In the second group, we have variables related with the COVID-19 such as the cumulative number of deaths, confirmed cases, test, etc. For the moment, we will select all of them and then we will decide. In the policy measures, we have very important variables for the development of the pandemic and for our study, like `workplace_closing`, `stay_home_restrictions` or `internal_movement_restrictions`, so we must take them into account. Finally, there are other geographical variables that could be interesting for our models. A description of our variables can be seen below:

Variable	**Description**
`id`	Unique identifier.
`date`	Observation date.
`deaths`	Cumulative number of deaths.
`test`	Cumulative number of test.
`confirmed`	Cumulative number of confirmed cases.
`vaccines`	Cumulative number of doses administered (single dose).
`hosp`	Number of hospitalized patients on date.
`icu`	Number of hospitalized patients in ICUs on date.
`population`	Total population.
`stay_home_restrictions`	Indicates the measures of staying at home.
`school_closing`	Indicates the measures in education.
`workplace_closing`	Indicates the measures of the workplace.
`transport_closing`	Indicates the measures in the public transport.
`gatherings_restrictions`	Indicates the measures of gatherings.
`internal_movement_restrictions`	Indicates the measures of the movements between regions.

We have realized that all numerical variables have **missing values**. Some of them like the deaths or confirmed cases have them at the beginning, but we have no problem with that because there were not any cases at that time, so we will give 0 to these values. However, there are other variables for which we have to think

what to do with their missing values very carefully. For instance, for the tests, there are 399 missing values out of 451 observations, so we will remove it because there is no much information. For the vaccines, we have 378, which we might expect that since the first day we had vaccines was the 3rd of January of 2021 but also, we have some of them in the weekends. For the number of hospitalized people and the ICU people, we have again 303 NAs, but for the moment they will remain in the data set. How to deal with these variables that have missing values will be seen in next sections. Finally, the variables for the measures do not have any missing value.

After that, we should modify our categorical variables, since they are encoded as integers:

- The first one, `stay_home_restrictions`, will have three categories: no measures, require not leaving house with exceptions (e.g: exercise) and not leaving the house.
- The next one, `school_closing` has another four categories: no measures, recommended closing, require closing for some schools and require closing.
- For `workplace_closing` we have no measures, recommended closing, require closing for some sectors and require closing.
- For `transport_closing` we have two levels: no measures and recommended closing.
- For `gatherings_restrictions`: no measures, restrictions between 10-100 people and restrictions on gatherings of less than 10 people.
- Finally, the variable `internal_movement_restrictions` has no measures, recommend closing or require closing.

After these changes, in the next section we will perform some visualizations to know the distributions of the variables, relationship between them, etc.

3 EXPLORATORY DATA ANALYSIS (EDA)

Our two main variables of interest are cumulative `deaths` and `confirmed` cases. First of all, let's see how they are distributed.

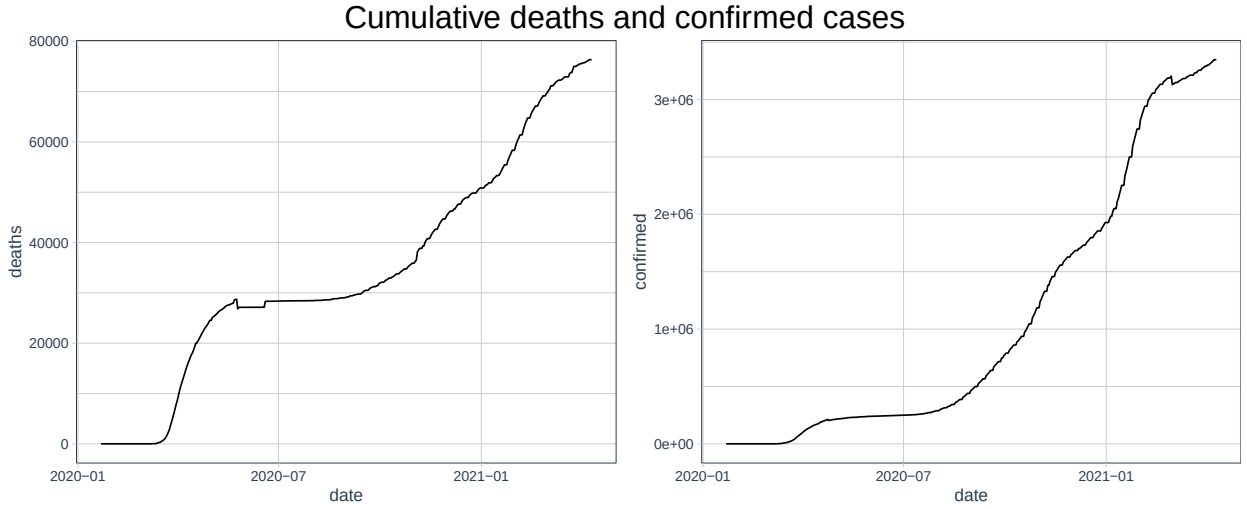


Figure 1: Cumulative deaths and confirmed cases in Spain

As we can observe, there are some mistakes in the data: in the cumulative deaths, there is a drop around June, and in the cumulative confirmed cases, there is another drop in February, which do not make any sense

since we are considering **cumulative** cases. Therefore, to fix that error, for all days that has a lower value than their previous day, we will assign the value of their previous day to that day.

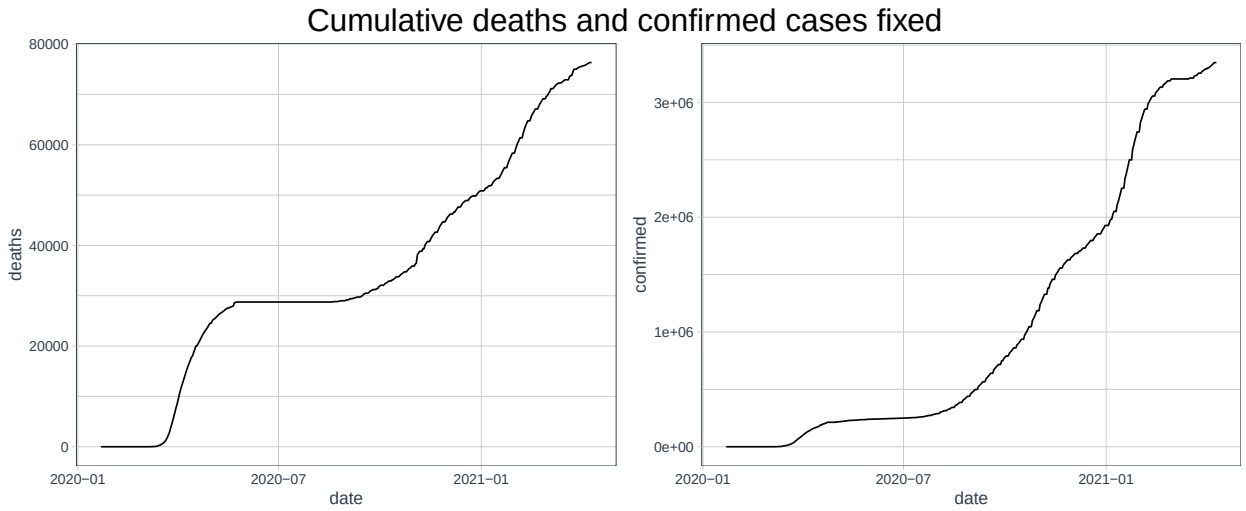


Figure 2: Cumulative deaths and confirmed cases fixed

Now let's focus on the vaccines administered and the hospitalized people:

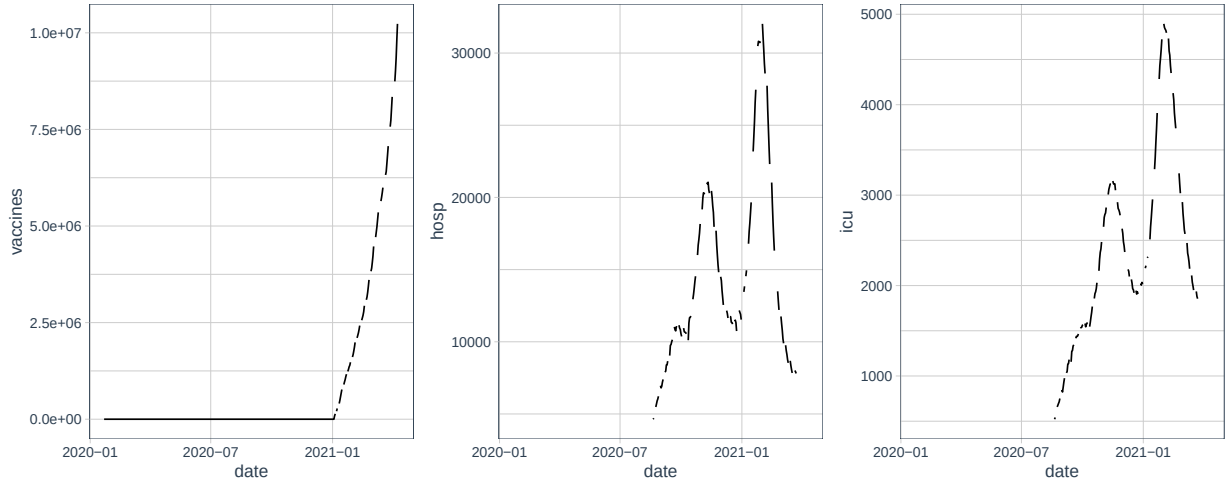


Figure 3: Cumulative vaccines, number of hospitalized people and number of patients in ICUs on date

Due to the missing values, we have discontinuous plots. For `vaccines`, these missing values appear because there are some places in Spain where the vaccination is stopped on weekends, or because these data is not recorded on weekends. Hence, as we have done before for the deaths and confirmed cases, we will take the value from the day before. However, for the other two variables, there is no information prior to July, so we will not take them into account because it is not realistic.

Then, we are going to see the relationships between our target variables, deaths and confirmed cases, and some of the categorical predictors. Below, we can see how predictors like `stay_home`, `workplace_closing`, `gatherings_restrictions` and `internal_movement_restrictions` are related with the confirmed cases:

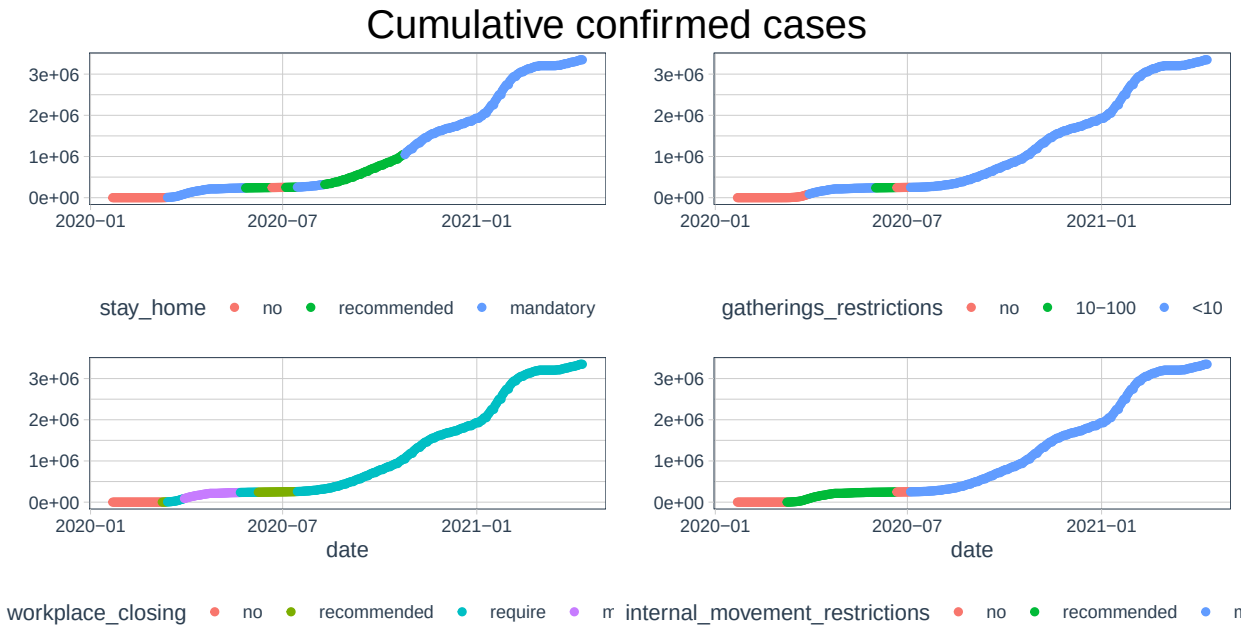


Figure 4: Relationship between the cumulative confirmed cases and some categorical variables

The first plot in the top left corner shows us that the variable `stay_home` is wrongly collected, since it was not mandatory to stay at home after the summer, so we must change it. Also, we can observe how after the summer the measures were much more restrictives than during the summer because of the increment of new cases. For the cummulative deaths, we can see it below:

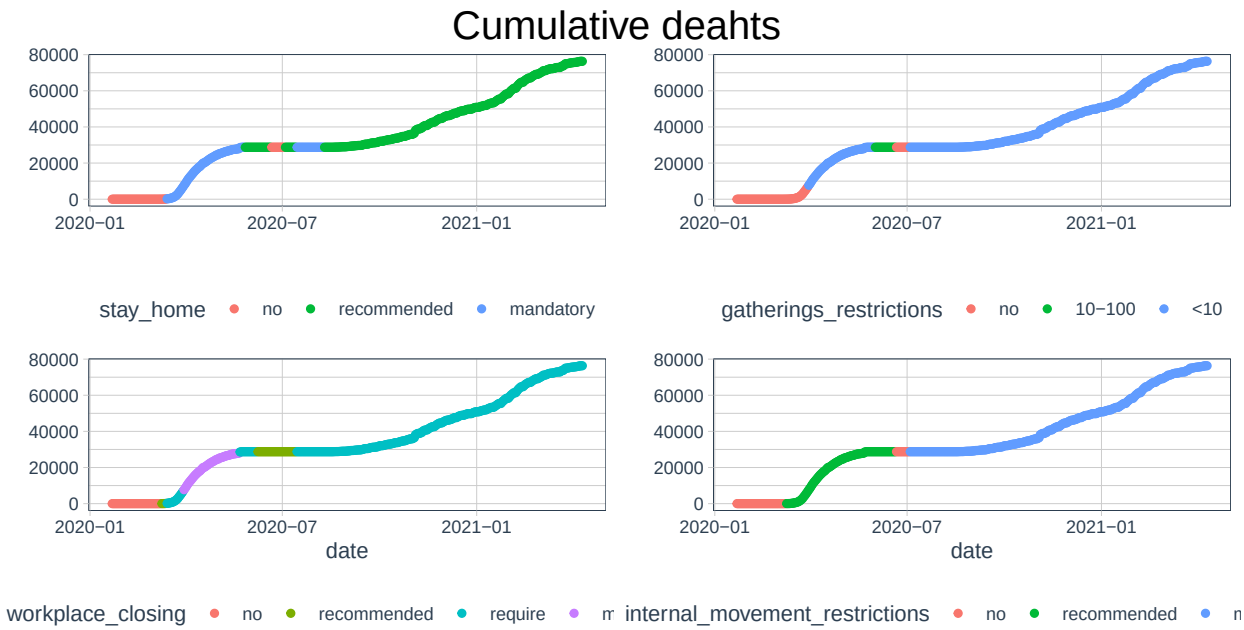


Figure 5: Relationship between the cumulative deaths and some categorical variables

Again, after summer the measures become more agressive due to the new cases. In the following section, we will see how to modify some variables after we have more knowledge about them.

4 FEATURE ENGINEERING

In this section, based on the previous analysis of the variables, we should decide which changes should be done in our data. Firslty, since our variables of interest, deaths and confirmed cases, are cumulative and daily, we will aggregate them to have the **number of deaths and confirmed cases by week**. This decision has been taken because there are too much noise in daily data due to different human errors that appear when this data is collected, so taking the data weekly allow us to predict in a better way the future.

On the one hand, to obtain the number of deaths, confirmed cases and vaccines per week, we will compute firstly the daily cases, substracting today's cases minus tomorrow's, and then we will add all the cases of the week. On the other hand, for the categorical variables, we will compute the mode of the variables in every week, i.e., we will give the most frequent values to every week.

Now that we have prepared our data, we have to think about what **preprocessing** steps should we do. These are some changes that we have to apply to our data before we start fitting the models. Our preprocessing steps will be:

1. Divide the date in month and year and convert it to categorical, because they could be very important for the outcomes.
2. Since we want to predict the deaths and confirmed cases in the following weeks, we should take into account how many deaths and confirmed cases were the weeks before. Therefore, we will create some new variables which will be the deaths and confirmed cases 3 weeks lagged, i.e., we will consider in our model what happened in the previous weeks to be able to predict in a better way the next month.
3. Before starting fitting models, we should take a look to our predictors. We have 12 covariates which the majority of them are categorical, so we must do **feature selection**. To do that, we will use the package `recipeselectors` (Pawley 2021), which adds some feature selection methods to the `recipes` package (from `tidymodels`) such as variable importance or boluta methods. This time we will use recursive feature elimination, selecting the features that are above the 20% of importance using a random forest model. For the model that predicts the deaths per week, it removes the variables `stay_home` and `gathering_restrictions`, nonetheless, for the confirmed cases, it removes `workplace_closing` and again `gathering_restrictions`.
4. Even though we will **test** our data with the next 3 weeks, we should divide our training data into folds to tune the model parameters and to compare them. To do that, there is a function called `rolling_origin` from the `rsample` package (from `tidymodels`) which can divide our training set into different folds by date, which is very useful for time series models. We will get 10 folds where the analysis set will be the first 12 weeks, and the assessment set, the following 3. This method will allow us to tune the different parameters of our models and get the best with the lowest prediction error. After selecting the two best ones, we will again fit the models in all our training set and then, we will see how it works in our test set.
5. Finally, it is time to use the `recipes` package (Kuhn and Wickham 2021) to do the preprocessing steps that the models need. For the linear regression models (`lm`, `ridge`, `lasso` and `elastic net`), the recommended preprocessing is to remove the zero variance variables, decorrelate the predictors and create dummy variables for the categorical predictors; and for the partial least squares (PLS), we need to normalize the predictors, i.e., transform them to have mean equal to 0 and variance equal to 1.

After defining all these steps, we are able to start creating our models. In the next section, we are going to explain the models we are going to analyze as well as all the steps that need to be followed to create a good model.

5 MODEL TUNING AND VALIDATION

As we have said in the previous section, we are going to try 6 different models:

- Simple and robust linear models (`lm` and `rlm`). Since the robust linear model is not implemented in the `parsnip` package, it has been created. In the script `rlm.R` there is the implementation of the model.
- Three different models from the `glmnet` package: ridge, lasso and elastic net regression. The difference between these three models is the parameter tuning. We have two parameters to tune: `penalty`, which represents the total amount of regularization, and `mixture`, which is a number between 0 and 1 that indicates the proportion of L1 regularization in the model. When `mixture = 0`, we get a ridge model but when `mixture = 1` we get a lasso model. When we decide to tune also the `mixture` parameter, then we get a elastic net model.
- A partial least squares model (`PLS`), when we have to tune two parameters: `predictor_prop`, which is the proportion of predictors that are allowed to affect each PLS component, and `num_comp`, which is the number of PLS components.

Thanks to the `workflowsets` package (Kuhn 2021), we can create two sets of **workflows** (pipelines in python), one for the deaths model and the other for the confirmed cases. A workflow is an object which is formed by a recipe and a model.

After defining these models, it is time to tune their parameters. With the R package `doParallel` (Corporation and Weston 2020), we are able to select the number of cores in order to decrease the computational time. In my computer, using 8 cores instead of 4 decreases more than half the computational time, which helps a lot. Next, we modify the intervals of the model parameters, and we will select the set of them that minimizes the root mean square error (`rmse`) and the ordinal r-square (`rsq`). Below, we can see the results for the deaths model:

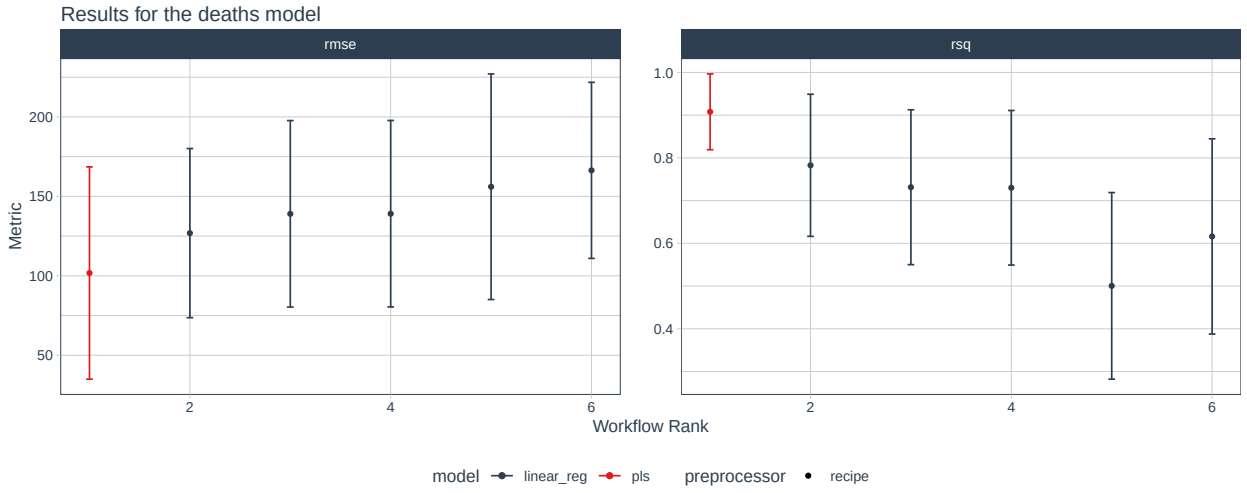


Figure 6: Results for the deaths models

As we can observe above, there is a big difference between the partial least squares model and the rest. The mean for the RMSE is around 100, i.e., the mean error is around 100 deaths, and the RSE is around 0.9. Both of these metrics have a very good results. In the same way, as we can see for the models of confirmed cases, the one with the best metrics is also the partial least squares, which its RMSE is below 10000 and the RSE is again close to 0.9.

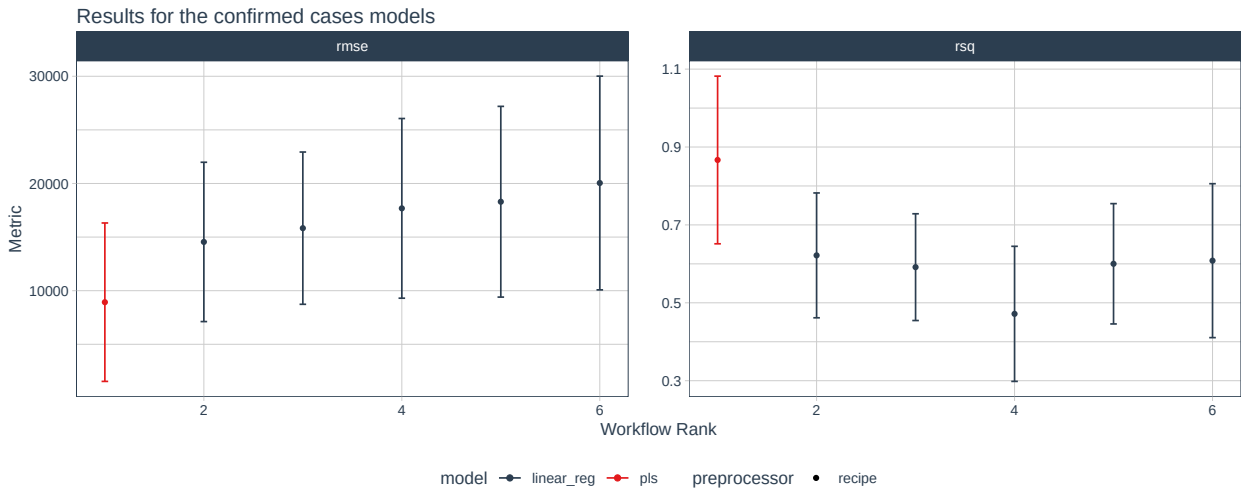


Figure 7: Results for the confirmed cases models

6 MODEL SELECTION AND CONCLUSIONS

As we have seen in the previous chapter, the best model for both cases is the **Partial Least Squares (PLS) regression**. As we briefly described before, it is a statistical method that is related with the principal components regression. It reduces the predictors to a smaller set of uncorrelated components and perform least squares regression with these components instead of using the original data. For both models, we can observe the final parameters:

Table 2: Final set of parameters for each model

Model	`predictor_prop`	`num_comp`
Deaths	0.1344	3
Confirmed cases	0.5607	2

The first parameter, `predictor_prop`, indicates the proportion of variables that are used in every PLS component. Since we have 24 predictors after the preprocessing, 4 and 13 variables are used on each of the sPLS components. The second parameter indicates the number of PLS components that are used as predictors in each model.

Consecutively, we have to *finalize* the workflows, i.e., we have to give the best set of parameters to each model. And now, we are able to fit the model **in all the training set**, since the previous folds were used only to get these parameters.

After fitting the models, in order to see some characteristics of them, we can plot a *Correlation Circle Plot*, which represents the two components and projects the variables in the plane defined by them. Strongly associated or correlated variables are plotted in the same direction. The greater the distance from the origin the stronger the association. For the deaths model, we can observe two main groups whose variables are very correlated with the outcome.

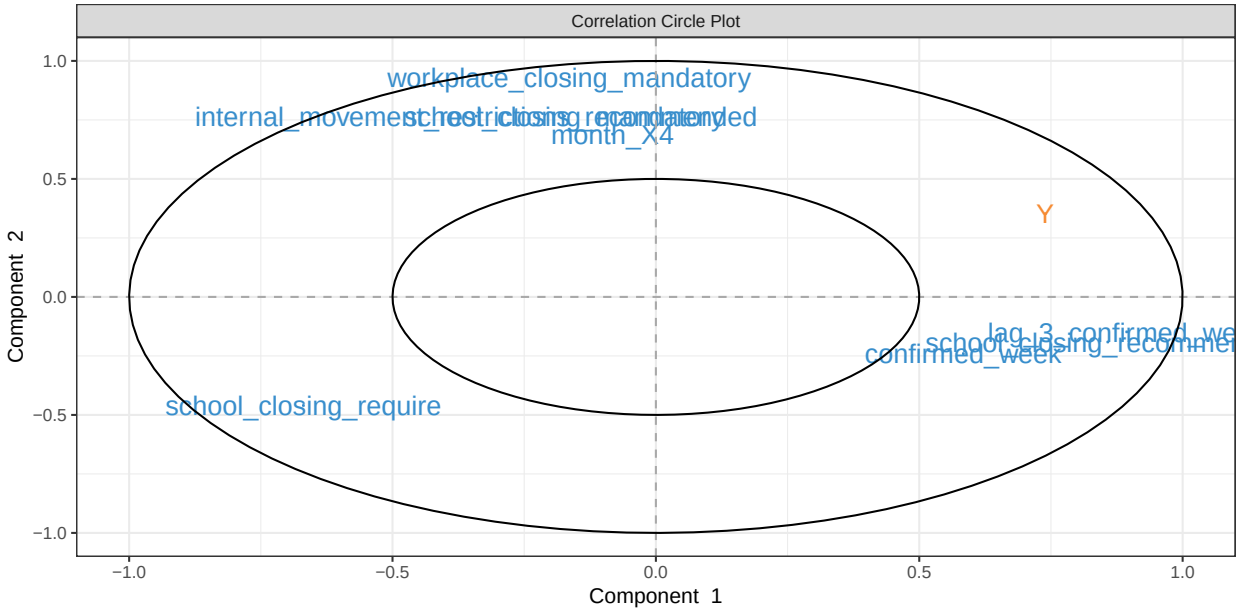
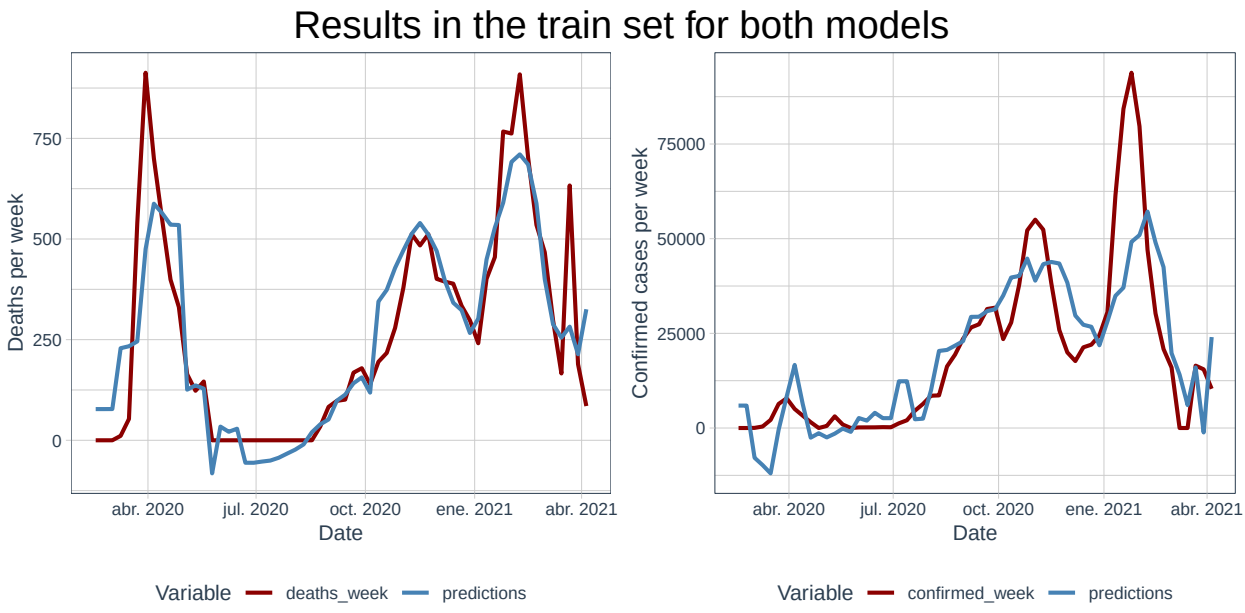


Figure 8: Correlation Circle Plot for the deaths model

After analyzing the characteristics of the best models, we are going to see the predictive power of the models in the training and testing sets. Below we can observe the difference between the observed and predicted values for both variables deaths and confirmed cases per week:



Even though the predictions are very similar to the real values, we cannot reach the peaks of deaths that appear in April or February, and the peak of confirmed cases of February. This happen because we want to predict the following three weeks, so we are considering 3 lagged variables. If we wanted to predict only the next week, we would consider the value of deaths and confirmed cases of the day before, so we would have a much more accurate model.

In the testing set we have the following results:

Table 3: Results in the testing set

Date	Deaths	Predicted Deaths	Difference Deaths	Confirmed	Predicted Confirmed	Difference Confirmed
2021-04-12	197	404	207	22744	29669	6925
2021-04-19	121	431	310	21071	14265	6806
2021-04-26	147	410	263	19852	25063	5211

REFERENCES

Corporation, Microsoft, and Steve Weston. 2020. *doParallel: Foreach Parallel Adaptor for the 'Parallel' Package*. https://CRAN.R-project.org/package=doParallel.

Guidotti, Emanuele, and David Ardia. 2020. "COVID-19 Data Hub." *Journal of Open Source Software* 5 (51): 2376. https://doi.org/10.21105/joss.02376.

Kuhn, Max. 2021. *Workflowsets: Create a Collection of 'Tidymodels' Workflows*. https://CRAN.R-project.org/package=workflowsets.

Kuhn, Max, and Hadley Wickham. 2020. *Tidymodels: A Collection of Packages for Modeling and Machine Learning Using Tidyverse Principles*. https://www.tidymodels.org.

———. 2021. *Recipes: Preprocessing Tools to Create Design Matrices*. https://CRAN.R-project.org/package=recipes.

Pawley, Steven. 2021. *Recipeselectors: Extra Recipes Steps for Supervised Feature Selection*. https://github.com/stevenpawley/recipeselectors.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. "Welcome to the tidyverse." *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.