Election Forecasting Revisited

Background Information on the Dataset

In the recitation from Unit 3, we used logistic regression on polling data in order to construct US presidential election predictions. We separated our data into a training set, containing data from 2004 and 2008 polls, and a test set, containing the data from 2012 polls. We then proceeded to develop a logistic regression model to forecast the 2012 US presidential election.

In this homework problem, we’ll revisit our logistic regression model from Unit 3, and learn how to plot the output on a map of the United States. Unlike what we did in the Crime lecture, this time we’ll be plotting predictions rather than data!

First, load the ggplot2, maps, and ggmap packages using the library function. All three packages should be installed on your computer from lecture, but if not, you may need to install them too using the install.packages function.

*# Load packages*

**library**(maps)

**library**(devtools)

**library**(ggmap)

**register\_google**(key = "AIzaSyBlCZXGDK9dN3Vf\_N1qdI6mPfFFCA34ubs")

Then, load the US map and save it to the variable statesMap, like we did during the Crime lecture:

*# Load StateMap*

statesMap = **map\_data**("state")

The maps package contains other built-in maps, including a US county map, a world map, and maps for France and Italy.

Drawing a Map of the US

If you look at the structure of the statesMap data frame using the str function, you should see that there are 6 variables. One of the variables, group, defines the different shapes or polygons on the map. Sometimes a state may have multiple groups, for example, if it includes islands.

*# Output structure*

**str**(statesMap)

## 'data.frame': 15537 obs. of 6 variables:

## $ long : num -87.5 -87.5 -87.5 -87.5 -87.6 ...

## $ lat : num 30.4 30.4 30.4 30.3 30.3 ...

## $ group : num 1 1 1 1 1 1 1 1 1 1 ...

## $ order : int 1 2 3 4 5 6 7 8 9 10 ...

## $ region : chr "alabama" "alabama" "alabama" "alabama" ...

## $ subregion: chr NA NA NA NA ...

How many different groups are there?

z = **table**(statesMap**$**group)

**kable**(z)

| **Var1** | **Freq** |
| --- | --- |
| 1 | 202 |
| 2 | 149 |
| 3 | 312 |
| 4 | 516 |
| 5 | 79 |
| 6 | 91 |
| 7 | 94 |
| 8 | 10 |
| 9 | 872 |
| 10 | 381 |
| 11 | 233 |
| 12 | 329 |
| 13 | 257 |
| 14 | 256 |
| 15 | 113 |
| 16 | 397 |
| 17 | 650 |
| 18 | 399 |
| 19 | 566 |
| 20 | 36 |
| 21 | 220 |
| 22 | 30 |
| 23 | 460 |
| 24 | 370 |
| 25 | 373 |
| 26 | 382 |
| 27 | 315 |
| 28 | 238 |
| 29 | 208 |
| 30 | 70 |
| 31 | 125 |
| 32 | 205 |
| 33 | 78 |
| 34 | 16 |
| 35 | 290 |
| 36 | 21 |
| 37 | 168 |
| 38 | 37 |
| 39 | 733 |
| 40 | 12 |
| 41 | 105 |
| 42 | 238 |
| 43 | 284 |
| 44 | 236 |
| 45 | 172 |
| 46 | 66 |
| 47 | 304 |
| 48 | 166 |
| 49 | 289 |
| 50 | 1088 |
| 51 | 59 |
| 52 | 129 |
| 53 | 96 |
| 54 | 15 |
| 55 | 623 |
| 56 | 17 |
| 57 | 17 |
| 58 | 19 |
| 59 | 44 |
| 60 | 448 |
| 61 | 373 |
| 62 | 388 |
| 63 | 68 |

There are 63 different groups.

Which one defined the color of the outline of the states?

*# Draw Map of US*

**ggplot**(statesMap, **aes**(x = long, y = lat, group = group)) **+** **geom\_polygon**(fill = "white", color = "black")

Diagram

Description automatically generated

Color = outline of the states

Coloring the States by Predictions

Now, let’s color the map of the US according to our 2012 US presidential election predictions from the Unit 3 Recitation. We’ll rebuild the model here, using the dataset PollingImputed.csv. Be sure to use this file so that you don’t have to redo the imputation to fill in the missing values, like we did in the Unit 3 Recitation.

Load the data using the read.csv function, and call it “polling”. Then split the data using the subset function into a training set called “Train” that has observations from 2004 and 2008, and a testing set called “Test” that has observations from 2012.

Note that we only have 45 states in our testing set, since we are missing observations for Alaska, Delaware, Alabama, Wyoming, and Vermont, so these states will not appear colored in our map.

Then, create a logistic regression model and make predictions on the test set using the following commands:

*# Read data*

polling = **read.csv**("PollingImputed.csv")

*# Split the data*

Train = **subset**(polling, Year **==** 2004 **|** Year **==** 2008)

Test = **subset**(polling, Year **==** 2012)

*# Logistic Regression*

mod2 = **glm**(Republican**~**SurveyUSA**+**DiffCount, data=Train, family="binomial")

*# Make predictions*

TestPrediction = **predict**(mod2, newdata=Test, type="response")

*# Vector of Republican/Democrat*

TestPredictionBinary = **as.numeric**(TestPrediction **>** 0.5)

*# Store predictions and state labels into a dataframe*

predictionDataFrame = **data.frame**(TestPrediction, TestPredictionBinary, Test**$**State)

For how many states is our binary prediction 1 (for 2012), corresponding to Republican?

*# Tabulate how many states is our prediction 1*

z = **table**(TestPredictionBinary)

**kable**(z)

| **TestPredictionBinary** | **Freq** |
| --- | --- |
| 0 | 23 |
| 1 | 22 |

22 states have a binary prediction of 1

What is the average predicted probability of our model (on the Test set, for 2012)?

*# Average predicted probability*

**summary**(TestPrediction)

## Min. 1st Qu. Median Mean 3rd Qu. Max.

## 0.0000001 0.0000926 0.0648667 0.4852626 0.9986385 0.9998655

Average Predicted Probability = 0.4852

Merge Data

Now, we need to merge “predictionDataFrame” with the map data “statesMap”, like we did in lecture. Before doing so, we need to convert the Test.State variable to lowercase, so that it matches the region variable in statesMap. Do this by typing the following in your R console:

*# PredictionBinary*

predictionDataFrame**$**region = **tolower**(predictionDataFrame**$**Test.State)

Now, merge the two data frames using the following command:

*# Merge the data*

predictionMap = **merge**(statesMap, predictionDataFrame, by = "region")

Lastly, we need to make sure the observations are in order so that the map is drawn properly, by typing the following:

*# Order the observations*

predictionMap = predictionMap[**order**(predictionMap**$**order),]

How many observations are there in predictionMap?

*# Number of observations*

**nrow**(predictionMap)

## [1] 15034

15034 observations.

How many observations are there in statesMap?

*# Number of observations*

**nrow**(statesMap)

## [1] 15537

15537 observations.

When we merged the data in the previous problem, it caused the number of observations to change. Why?

When we merge data, it only merged the observations that exist in both data sets. So since we are merging based on the region variable, we will lose all observations that have a value of “region” that doesn’t exist in both data frames. You can change this default behavior by using the all.x and all.y arguments of the merge function.

Color the US map with our predictions

You can color the states according to our binary predictions by typing the following in your R console:

*# Color the US Map*

**ggplot**(predictionMap, **aes**(x = long, y = lat, group = group, fill = TestPredictionBinary)) **+** **geom\_polygon**(color = "black")

A picture containing map

Description automatically generated

The states appear light blue and dark blue in this map. Which color represents a Republican prediction?

Our logistic regression model assigned 1 to Republican and 0 to Democrat. As we can see from the legend, 1 corresponds to a light blue color on the map and 0 corresponds to a dark blue color on the map.

Replot with discrete outcomes

We see that the legend displays a blue gradient for outcomes between 0 and 1. However, when plotting the binary predictions there are only two possible outcomes: 0 or 1. Let’s replot the map with discrete outcomes. We can also change the color scheme to blue and red, to match the blue color associated with the Democratic Party in the US and the red color associated with the Republican Party in the US. This can be done with the following command:

*# Replot to show discrete*

**ggplot**(predictionMap, **aes**(x = long, y = lat, group = group, fill = TestPredictionBinary))**+** **geom\_polygon**(color = "black") **+** **scale\_fill\_gradient**(low = "blue", high = "red", guide = "legend", breaks= **c**(0,1), labels = **c**("Democrat", "Republican"), name = "Prediction 2012")

Map

Description automatically generated

Alternatively, we could plot the probabilities instead of the binary predictions. Change the plot command above to instead color the states by the variable TestPrediction. You should see a gradient of colors ranging from red to blue.

*# Replot to show discrete*

**ggplot**(predictionMap, **aes**(x = long, y = lat, group = group, fill = TestPrediction))**+** **geom\_polygon**(color = "black") **+** **scale\_fill\_gradient**(low = "blue", high = "red", guide = "legend", breaks= **c**(0,1), labels = **c**("Democrat", "Republican"), name = "Prediction 2012")

Map

Description automatically generated

Do the colors of the states in the map for TestPrediction look different from the colors of the states in the map with TestPredictionBinary? Why or why not?

The two maps look very similar. This is because most of our predicted probabilities are close to 0 or close to 1.

Understanding the Predictions

In the 2012 election, the state of Florida ended up being a very close race. It was ultimately won by the Democratic party.

Did we predict this state correctly or incorrectly?

We incorrectly predicted this state by predicting that it would be won by the Republican party.

What was our predicted probability for the state of Florida?

z = predictionDataFrame

**kable**(z)

|  | **TestPrediction** | **TestPredictionBinary** | **Test.State** | **region** |
| --- | --- | --- | --- | --- |
| 7 | 0.9739028 | 1 | Arizona | arizona |
| 10 | 0.9994949 | 1 | Arkansas | arkansas |
| 13 | 0.0000926 | 0 | California | california |
| 16 | 0.0094330 | 0 | Colorado | colorado |
| 19 | 0.0000343 | 0 | Connecticut | connecticut |
| 24 | 0.9640395 | 1 | Florida | florida |
| 27 | 0.9901680 | 1 | Georgia | georgia |
| 30 | 0.0000478 | 0 | Hawaii | hawaii |
| 33 | 0.9996372 | 1 | Idaho | idaho |
| 36 | 0.0000926 | 0 | Illinois | illinois |
| 39 | 0.9992970 | 1 | Indiana | indiana |
| 42 | 0.0648667 | 0 | Iowa | iowa |
| 45 | 0.9506137 | 1 | Kansas | kansas |
| 48 | 0.9901659 | 1 | Kentucky | kentucky |
| 51 | 0.9994949 | 1 | Louisiana | louisiana |
| 54 | 0.0009383 | 0 | Maine | maine |
| 57 | 0.0000024 | 0 | Maryland | maryland |
| 60 | 0.0000001 | 0 | Massachusetts | massachusetts |
| 63 | 0.0000177 | 0 | Michigan | michigan |
| 66 | 0.0004843 | 0 | Minnesota | minnesota |
| 69 | 0.9325489 | 1 | Mississippi | mississippi |
| 72 | 0.9990219 | 1 | Missouri | missouri |
| 75 | 0.9986385 | 1 | Montana | montana |
| 78 | 0.9998655 | 1 | Nebraska | nebraska |
| 81 | 0.0001795 | 0 | Nevada | nevada |
| 84 | 0.0000665 | 0 | New Hampshire | new hampshire |
| 87 | 0.0000127 | 0 | New Jersey | new jersey |
| 90 | 0.0018172 | 0 | New Mexico | new mexico |
| 93 | 0.0000013 | 0 | New York | new york |
| 96 | 0.9506205 | 1 | North Carolina | north carolina |
| 99 | 0.9998655 | 1 | North Dakota | north dakota |
| 102 | 0.0000024 | 0 | Ohio | ohio |
| 105 | 0.9996372 | 1 | Oklahoma | oklahoma |
| 108 | 0.0035166 | 0 | Oregon | oregon |
| 111 | 0.0000926 | 0 | Pennsylvania | pennsylvania |
| 114 | 0.0004844 | 0 | Rhode Island | rhode island |
| 117 | 0.9994949 | 1 | South Carolina | south carolina |
| 120 | 0.9949023 | 1 | South Dakota | south dakota |
| 123 | 0.9996372 | 1 | Tennessee | tennessee |
| 126 | 0.9973641 | 1 | Texas | texas |
| 129 | 0.9992969 | 1 | Utah | utah |
| 134 | 0.0181252 | 0 | Virginia | virginia |
| 137 | 0.0000246 | 0 | Washington | washington |
| 140 | 0.9981049 | 1 | West Virginia | west virginia |
| 143 | 0.0006740 | 0 | Wisconsin | wisconsin |

Predicted Probability = 0.9640395

What does this imply?

Our prediction model did not do a very good job of correctly predicting the state of Florida, and we were very confident in our incorrect prediction.

Parameter Settings

In this part, we’ll explore what the different parameter settings of geom\_polygon do. Throughout the problem, use the help page for geom\_polygon, which can be accessed by ?geom\_polygon. To see more information about a certain parameter, just type a question mark and then the parameter name to get the help page for that parameter. Experiment with different parameter settings to try and replicate the plots!

We’ll be asking questions about the following three plots:

Map

Description automatically generated with low confidence

A picture containing diagram

Description automatically generated

Map

Description automatically generated

*# Plot 1*

**ggplot**(predictionMap, **aes**(x = long, y = lat, group = group, fill = TestPrediction))**+** **geom\_polygon**(color = "black", linetype=3) **+** **scale\_fill\_gradient**(low = "blue", high = "red", guide = "legend", breaks= **c**(0,1), labels = **c**("Democrat", "Republican"), name = "Prediction 2012")

Map

Description automatically generated

*# Plot 2*

**ggplot**(predictionMap, **aes**(x = long, y = lat, group = group, fill = TestPrediction))**+** **geom\_polygon**(color = "black", size=3) **+** **scale\_fill\_gradient**(low = "blue", high = "red", guide = "legend", breaks= **c**(0,1), labels = **c**("Democrat", "Republican"), name = "Prediction 2012")

A picture containing diagram

Description automatically generated

*# Plot 3*

**ggplot**(predictionMap, **aes**(x = long, y = lat, group = group, fill = TestPrediction))**+** **geom\_polygon**(color = "black", alpha=0.3) **+** **scale\_fill\_gradient**(low = "blue", high = "red", guide = "legend", breaks= **c**(0,1), labels = **c**("Democrat", "Republican"), name = "Prediction 2012")

Chart

Description automatically generated

What is the name of the parameter we changed to create plot (1)?

linetype

What is the name of the parameter we changed to create plot (2)?

size

Plot (3) was created by changing the value of a different geom\_polygon parameter to have value 0.3. Which parameter did we use?

alpha