Week 10 Problem Set: Multiple regression and GLMs

(30 pts)

Part I: Warm up and review

Fill out the following table (0.5 point each cell = 6 points total)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Hypothesis  (assuming two-tailed tests) | Test statistic T | f(T|H0) (Distribution of T under H0) | Assumptions |
| Pearson’s product moment correlation |  |  |  |  |
| Spearman’s rank correlation |  |  |  |  |
| Kendall’s coefficient of rank correlation |  |  |  |  |

Part II

Load the dataset “NYData.csv”. These data represent number of **new** confirmed coronavirus cases in the first days after March 1, 2020 in the state of New York. (March 1st is Day=1.) There are 19.54 million residents in the state of New York.

(A) Plot the number of cumulative cases as a function of Day. COPY AND PASTE YOUR SCATTERPLOT BELOW (2 pt). Use all of the guidelines for effective display of data discussed in Week 7.

(B) Fit the cumulative coronavirus data using logistic regression, and plot the best fit line on the original data. Using R’s function ‘predict.glm’, plot the 95th percentile confidence intervals as well (we can assume the errors are normally distributed and use ±1.96 s.e.). COPY AND PASTE YOUR MODEL FIT, HOLD OFF ON PLOT UNTIL [D]). Note that a logistic regression is used when the underlying data follow a Binomial process. Think carefully about how to set up your logistic regression so it reflects the Binomial nature of the data you have. (5 pt)

(C) (10 pts total) We will now fit an alternative model to these data, one that is often used for processes like epidemic spread. Fit the Gompertz curve

Note that this is equivalent to:

to the coronavirus data. You might start trying R’s simple function ‘nls’ but you might find that it is hard to find starting values that are close enough for ‘nls’ to actually converge on estimates. One trick is to take the log of the above equation (log both the left and right hand sides) and then try fitting that logged equation. This logged version is very similar (but not exactly the same) to the original question, and the estimates for this logged version will be good starting values. (Taking the log of both sides yields an equation which is numerically easier for ‘nls’ to handle.) If this doesn’t work for you, you might also try the ‘nlsLM’ function in the ‘minpack.lm’ package. More details here: http://www.r-bloggers.com/a-better-nls/.

COPY AND PASTE YOUR MODEL FIT, HOLD OFF ON PLOT UNTIL (D). Write a script to construct the 95th percentile confidence intervals using the following bootstrap technique (this is a bit of a hack, but one is often left with no choice but to get creative…):

1. sample with replacement from the residuals of the original ‘nls’ fit

2. construct a new bootstrap dataset using (Note that is a legitimate residual from the original fit, its just been moved to a new data point.)

3. refit the model as above

4. use nls() to obtain estimates for the model parameters ( is an index for which bootstrap sample you are on) ( using the data created by the bootstrapping procedure

5. save the model predictions for each x (i.e. )

6. repeat steps 1-5 1000 times (this will generate 1000 bootstrap replications)

7. for each value of x (i.e. Day), define the 95th percentile CI for (at that x) as the (2.5th, 97.5th) percentiles of the bootstrapped (at that x)

(Notice that your CI for will no longer be a smooth function! Also, you might find that inside your bootstrap loop, you are getting errors and this is causing the loop to abort with an error. This is happening because there will be times when, by random chance, the bootstrapped datasets will be hard to fit, and nls() will fail to converge. To avoid having an error on one iteration cause the whole loop to abort with an error, wrap the entire line of code with nls in the try() function. This will prevent an error from causing the loop to abort with an error. For more details, look at the help file ?try.)

PASTE SCRIPT BELOW:

(D) Plot the two best-fit curves on top of the plot from (A) and their 95th percentile confidence intervals. COPY AND PASTE YOUR PLOT BELOW. Once again, use all of the guidelines for effective display of data discussed in Week 7. (3 pt)

(E) Which model fits better and why? (There may be more than one correct answer depending on the metric used; I’m just interested in seeing you justify your decision in a reasonable way.) (3 pt)

(F) What is the difference between the interpretation of a confidence interval and that of a prediction interval? (1 pt)

**From the Slack archives:**

If your CI are super narrow, you haven't done it wrong. This comes about due to a combination of two factors, which I will list in decreasing order of relevance (in terms of their importance for making the CI narrow): 1) We are only looking at the very beginning of this curve; in percentage terms, only a tiny fraction of all NY residents are infected so its at the beginning of the logistic upswing. Plotting the predicted curve all the way to its asymptote (~200 days) shows that a bit better. 2) The data set is sort of large, with relatively little scatter around the line, so the model is highly certain about the model parameters. Jittering the data will help a bit, but since we are still at the beginning of the upswing, it helps less than you might think. If you were fitting a part of the curve where probability of infection was closer to 50%, then you would see the CI diverge a little more. This is a nice illustration of the variance of the Binomial model -> zero near the bounds of 0 and 1, so pretty much however bad the model fit is, the CI are still super narrow near 0, which is where are data are because the percentage of infections is super small. So that is why jittering the data (as an exploratory tool to see how much noise might increase the width of the CI and the model uncertainty) didn't help as much as it usually does. The real issue is that we are so close to probability ~0 that the uncertainty on is close to 0 irrespective of how lousy the model is (that is, even when we take the data, add noise, and downsample to just a few points).

Some of you may get errors when you try and use predict.glm to find predicted values for new data. The issue probably has to do with a mis-match in the names of the "Day" variable between when you fit the model and when you created the new data data.frame. In particular, many of you are probably referring to the covariate as NYData$Day when you fit the model but then are creating a new variable

Day<-data.frame()

for the prediction. R doesn't know that NYData$Day and Day are supposed to refer to the same thing. By far the easiest way to avoid this error is to create new variables from the columns of NYData when you read it in.

A helpful discussion on this error can be found here: <https://stackoverflow.com/questions/27464893/getting-warning-newdata-had-1-row-but-variables-found-have-32-rows-on-pred>