Pharmaceutical Anomalies Project

Michael Barrett

Overview

Tasked with finding anomalies in pharmaceutical spending/ordering, I decided that I would systematically create a regression for each drug in my dataset. The dependent variable is the amount spent on the specific drug, and the independent variables would be the number of patients with diagnosis *d*, and the accumulation of the length of stay for all patients with diagnosis *d*, with *d* being each and every DRG in the dataset. These regressions were created on a *by physician* basis, meaning the model would look like the following:

*ŷ The estimate for a physician’s total spending on the specific drug*

*The number of patients the physician had encountered of diagnosis* k

*The cumulative length of stay for all patients with diagnosis* k *encountered by the physician*

*The constant for the intercept*

*The set of constants generated by the least-squares method of linear regression*

Creating this regression would allow me to generate estimates for how much a physician may spend on a specific drug given all the independent variables. Comparing this estimate with the actual amount spent would allow me to determine how far off from the estimate this physician was, and whether that difference is extreme enough to be considered an anomaly.

Considerations

I decided that creating as estimate for expected spending on a drug would be critical, as it is difficult to see if a physician is spending an abnormal amount on said drug without accounting for a variety of variables, including the number of patients treated of each diagnosis, and the cumulative length of stay for all patients treated of each diagnosis. To create the estimate, I decided to go with a linear model because of its simplicity. The independent variables were chosen because they were relevant and could be recycled for the regression on each drug. Not every independent variable has a significant impact on each drug, and the relevant constant would reflect that by being insignificant. I chose total cumulative length of stay for patients of each diagnosis instead of average length of stay because I am measuring total costs, not average costs. For instance, a physician who sees many patients who each stay for less than a day would likely spend more on a relevant drug than a physician who only saw one patient with a length of stay of two days. I decided to not set the intercept, and to allow for an intercept term, so that the model would be more accurate.

Initially, I thought that I might look for anomalies on a *per department* basis, determining which departments at which hospitals had abnormal spending. Looking at the dataset, this idea was discarded because of how often patients switched departments. Many patients had a different admitting department than discharging department, and the department which ordered the drug was not even listed on the order. The physician, however, was. In addition to that, there were enough physicians that the sample size would be much larger than that of the departments. So I decided to look at anomalies on a *per physician* basis.

I then considered how I would go about creating these regressions for my estimates. Since the dataset I was given was a spreadsheet consisting of one order of one drug by a specific physician for a specific patient per line, so I knew that to analyze the data, I would first need to re-arrange it to fit the desired model. I would need to take each line off the dataset, and attribute the order to the appropriate physician. To do this, I had options. I could write a macro in Microsoft Excel or Microsoft Access, I could try to figure it out with Minitab, I could write code in MATLAB, I could write code in R, or I could look into using QlikSense. Having experience in MATLAB and Minitab, I did consider those. MATLAB is expensive, and I probably couldn’t justify it over the alternatives, and Minitab just didn’t seem like the tool for the job (the organizing bit, at least). Having no experience with Microsoft Visual Basic, I decided Excel and Access would be a last resort. From what I could tell, QlikSense was more geared towards creating dashboards to visualize the data, but not towards any type of analysis. Having access to it, I decided I would use it to visualize my results.

R seemed to fit the bill for the analysis. It is geared towards statistical analysis and is often used by professionals for large data. It is similar to MATLAB, so learning it would not be as hard as learning to write macros in Visual Basic. The capability of the language is more than sufficient for what I wanted to do, and there are many third-party packages that cover much more, should there be the need. I have been wanting to become proficient in R, but had I never acted on that ambition. It would certainly be a skill I could use in the future. Most importantly, it’s free. Not just free to download and install, but free to use in any way, and open source. It is published under the GNU Public License which defines your rights of use for this software. R does have drawbacks, however. It performs certain tasks slowly, support is entirely by volunteers (meaning it could be inconsistently written and have holes in certain topics), it struggles with managing large amounts of memory, and it is incapable of reading Microsoft Excel spreadsheets. I decided to go with R and put up with these drawbacks.

Writing the Code

Before this, I had only ever played in R, not doing and serious work. Through much trial and error, I was able to write a script that did what I wanted with poor optimization. It read the dataset from the CSV file, created a blank data frame for re-organizing the patient data to a *by physician* basis and a data frame for storing the total costs on each drug for each physician. It then went line by line of the original dataset, adding patient data to the appropriate physician in the data frame, if there were no missing values on that line in the dataset. Each patient that a physician prescribes a drug for is considered a patient treated by that physician. However, each stay of a treated patient is only attributed to each physician once, so if the physician sends two orders for the same patient (during the same stay), that patient’s data is only added to the physician’s row once. However, a patient may be attributed to more than one physician, if more than one orders a drug for said patient during the stay. After the appropriate data had been added (or skipped if it had previously been added) to the patient data data frame from the line in the dataset, the cost data was added to the appropriate cell in the cost data data frame, organized by ordering provider and drug prescribed, again, only if there were no missing values on that line. Regardless of whether the patient data from that line of the dataset was skipped or added, cost data was always added. This process is what re-organized the data to an analysis-friendly format.

The analysis part of the code would systematically create linear regressions to estimate the amount a physician would spend on a specific drug based on the patient data attributed to the physician. This was done using the Ordinary Least Squares method. Briefly, how this works is a coefficient is generated for each dependent variable such that the sum of the squared error term is minimized. Consider the following:

*The total cost for the drug in question for physician* j

*The patient data for physician* j

*The coefficient for variable* k

*The error term associated with physician* j

*m The total number of physicians*

*n The total number of independent variables*

*The term to be minimized by choosing correct coefficients*

There is good documentation on this method available, so I will not go into how the coefficients are chosen to minimize the sum of the squared error terms, nor will I describe how this method is accepted as appropriate for creating regressions that make predictions. Suffice it to say that R has a built in function that generates the coefficients, predictions, and error terms, as well as other descriptive data on the regression, such as r2 (the fraction of variance in the dependent variables described by the regression) and σ2 (the standard deviation of the error terms). The code would run a regression for each drug, considering each physician as a data point, and store each regression for later review.

Optimizing the Code

The code described above was written and tested to work with a small sample of the larger dataset, containing 200 lines of orders. When I tested the original code on the full dataset, containing 7.86 million lines of orders, it was slow and prone to crash. I began looking for functions that would minimize the time taken to read and assign data from the dataset, better manage the limited memory available on my computer, and fix any other bugs that caused instability. I found the *data.table* library, which contained functions that sped up reading the dataset and re-assigning values in a data frame. It also introduced the object class “data table” which is more efficient in terms of memory usage than “data frame” object. I also found the *compiler* library, which allowed me to compile often-used functions for quicker execution. I included progress bars to display visual feedback on the progress of the program.

Finally I re-wrote the code to view the dataset in chunks, each chunk containing 100,000 lines of pharmaceutical orders. This allowed the program to save the patient data and cost data data tables after assigning the data from each chunk, and better display where in the execution the code was unstable. I found that the code was crashing because it still wasn’t properly allocating the memory, and R was using more memory than my system had available. This was solved by again re-writing the code to change how it assigned lines of orders from the dataset to the patient and cost data data tables. The re-written code took a subset of the dataset consisting of all the lines to be attributed to a specific physician, extracted the pertinent data from the subset, then adding it to the patient data and cost data data tables. This was done for each physician. This method was notably slower than the previous ones employed, but it was stable.

Only then was I able to move on and test the second part of the code where the regressions were created. Again, I was met with the problem of R needing more memory than was available on my machine. My initial solution consisted of writing the regressions to the hard drive and erasing them from the system memory. This is another slower method, but it seemed like the next method to try. After letting it run with that method, the system memory didn’t skyrocket, but my hard drive soon filled up. I was unaware of how much data was in a single regression. It turns out that in each regression was stored all the information used to make it, meaning the patient data data table that I recycled for each regression was stored with each regression. That data table had information on each diagnosis for each physician. Since this data table is the same for each regression, I looked for a way for the function to skip storing that data in each regression. The only way I found would also skip storing other useful data in each regression, including r2 and σ2. I decided to use this method until I decided that I would need the other useful information, and see what I could do with just the basic information.

Analyzing the Regressions

The main goal of this project was initially to find anomalies in pharmaceutical spending. Analysis of the results of my program would find them (possibly using a second program). Each regression would predict an amount to spend based on the physician’s patient data, and that prediction could be compared to the actual value. A basic way of analyzing the regressions would be to find the difference of the actual value and the predicted value. These are the error terms of the regression, or the residuals. The greatest residuals could be looked at as anomalies. Perhaps a more refined method of analyzing the regression would be to look at the ratio of the residual over the predicted value. The greatest ratio would have the greatest residual in terms of its prediction, and could be considered an anomaly. I had an even more refined method for finding anomalies. I wanted to create normalized z-scores for each residual. This would allow me to look at all the residuals on an even basis.

Normalizing the residuals means dividing each residual by a standard deviation or standard error. The standard deviation/error is created by looking at the magnitude and the accuracy of the prediction. This value varies in each prediction, even in the same regression. The worse the prediction and the larger the magnitude of the prediction, the larger the standard deviation/error. The larger the standard deviation/error and the smaller the residual, the smaller the normalized z-score. That means that each prediction is held to its own standard, and less accurate predictions can be accounted for. I therefore think that the extreme z-scores are what should be considered anomalies.

The challenge in this is determining a method for calculating the standard deviation/error. I would need information on the regression that I decided to omit due to size constraints. This would mean re-writing the second part of the code, again, this time making sure that I save all the important information unique to each regression, and nothing more. To determine what information I need to calculate the standard deviation/error, I first needed to choose how to calculate the value. One method built in to R will calculate them automatically, however this method is extremely slow, as it also performs many unnecessary computations to give a complete analysis of the regression. To speed up the program, I decided to write my own program for finding the standard deviation/error. Finding a formula to work with is difficult. I found the following equation online:

*Standard error of the residual of data entry* j *in the model*

*Variance of all residuals in the model*

*m Total number of physicians (data entries)*

*Vector of dependent variables for data entry* j

*Vector of the averages of each dependent variable*

This method worked quicker, but I am not sure how exactly it works. For all the documentation available on the Ordinary Least Squares method, there is relatively little on finding the standard error of the residual. Moreover, this method gave results that conflict with the method built into R. I have no way seeing how the values generated by the method built into R are computed, so I do not know if they are exactly what I am looking for. Therefore, I took what I knew about variances and used that knowledge to make my own. My formula reads as follows:

*The standard error of the residual of a data entry*

*The standard error of the* kth *coefficient (generated with the regression)*

*The independent variables of the data entry*

*n The total number of independent variables*

Again, this method matched neither of the previous methods, but I understand how the standard errors of the coefficients linearly propagate to the total standard error.

Adjusting the Code

After thinking about which method I should choose to implement in the code, I decided not to choose, add each of the three methods, and let it take however long it needed to run. Once implemented, part II alone took 1.32 days to complete with the whole dataset. Let’s not get ahead of ourselves, though. I first needed to figure out a way to get all the information I needed from the regression without filling up my memory or hard drive. I decided to generate each regression in its entirety, and extract the output that I needed by copying and pasting to a new object, and then delete the regression. This method was less than ideal because it took time, the new object was more difficult to work with, and the mechanics of copying and pasting were difficult to work out.

For each drug, I still needed to create a regression, one that creates all the output possible. I then would extract the fitted costs (the predictions created by the regression), the residuals, the coefficients created by the regression, the standard error of each coefficient, and the R-squared value. Then, the code would predict the standard error three times, using three methods, based on what the computer has generated to this point. The first method was the built-in function that R can use, but I am unsure of how it works. The second method utilizes the function that I found online. The third method is the function I created. Finally, all the data (fitted costs, residuals, coefficients, standard errors of coefficients, and standard errors of the fitted data generated by methods I, II, and III) was saved to a file named after the drug so it could be easily identified and loaded to another R session. The saved data, as well as all the regression’s output (which is not saved), gets removed from the current session so there is room for the next regression.

After working out all the bugs, and running this program over a weekend, I did some preliminary analysis (dividing the residuals by each of the three standard errors and finding the maximum) on the column that represented total costs per physician. I found that the three different methods pointed to three different physicians. I wanted to ask about which one may be the best example, so I could focus on that method, but that conversation didn’t go as planned. I was informed that one of the most useful statistics is the cost per patient day, and that I should be using that in my search. After long, hard thought, I decided that I could not find a meaningful way to incorporate this statistic into the program that I had written until this point, so I decided to wrap up this project with a write up so that I could focus on a new program. Little progress was made on that due to limited time, but that is not the purpose of this write-up.

Conclusion

This program is far from perfect. It fails to account for multiple physicians treating a single patient, who will each spend less than the physician who treated a patient by himself. There is also no way to measure that if patients who are sicker are cared for by one physician, and healthier ones are treated by another, as the patients are only separated by diagnosis, not severity. On a similar note, if patients who have other illnesses than the recorded diagnosis all choose one physician over another. There are multiple things that could point to anomalies that are justified, or nonexistent. Also, I considered incorporating into each regression not only patient data, but also the amount spent on each other drug, to see if there are any correlations between prescribing drug A and prescribing drug B. A negative number could mean they are substitutes for one another, which may be good to know.

Each case may be different in medicine, and I know that there is no way to perfectly predict what it will take to reach a point that we can discharge a patient, especially with the data I have. However, if we can recognize general trends and analyze them, perhaps by looking at the variation in the trend based on different factors, then perhaps we can utilize them to a certain extent.