**Capstone Project Proposal**

**Sam Cannon**

**Business Problem**

I work for a large private medical practice located in Mississippi that has just begun to realize the potential of the data they have access to. A program was created at this practice in 2016 to begin administering allergy medications in the form of shots to patients who were coming to the clinic. These shots are administered by nurses who are scheduled to be at the clinic during the hours of operation (7:30 AM - 4:30 PM). Up to this point, the nurses have been staffed arbitrarily, with no attention being paid to the number of nurses scheduled on a given day. Scheduling the nurses in shifts in this manner has led to several problems, primarily:

* Nurses are over-scheduled, meaning there are too many nurses on staff during these hours, leading to loss in revenue
* Nurses are under-staffed, patients are not receiving the best care because the nursing staff is not prepared to handle a larger influx of patients

**The Data**

Data for this problem will be drawn from a Microsoft SQL Server over a period dating back to the first of January, 2016, when the new program was started. The data is a series of daily aggregated shot numbers, timestamped on the daily level, running up through the current day. I will pull the data from the SQL database each time I am going to train a new model to make sure that the model is receiving the most up-to-date data.

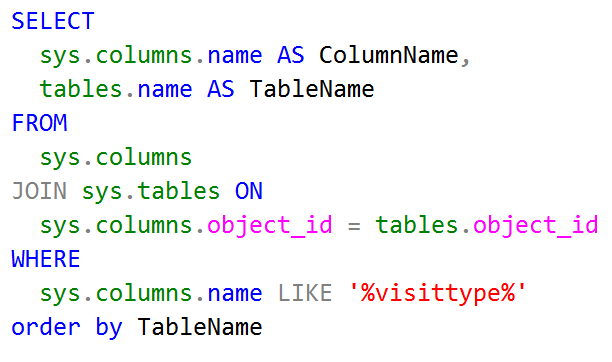
**Practical Implications**

Solving this problem will potentially allow the clinic to schedule the appropriate number of nurses per day. Scheduling the correct number of nurses will increase the quality of patient care while simultaneously reducing the cost of having nurses staffed who are unnecessary given the lower volumes of shots for any given day. Since the clinic is currently not basing their decisions off of any real data, this model will help introduce the possibilities of a data-driven approach to other business problems in addition to the current issue.

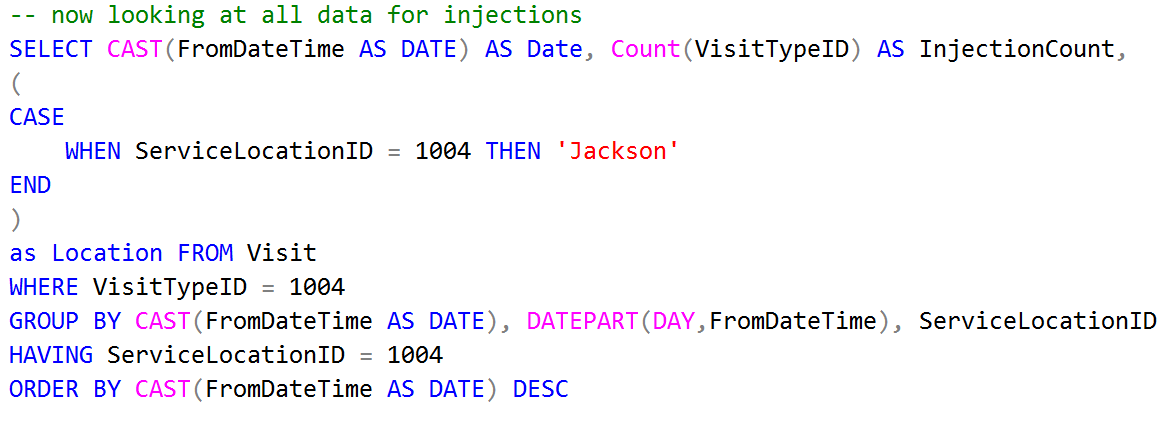
**Data Wrangling**

While the problem seemed very straightforward, accessing and getting the data into a usable format involved a few steps, namely locating the correct data in my company’s massive SQL database and exporting it to Python.

While it may seem simple, pulling data from my company’s SQL database has been a particularly difficult experience. Since my company has never used their data before, there are few established data routes to tables that are used frequently, and the company that designed the database did so out of ease of use for them, not ease of use for the company and someone who is unfamiliar with medical and billing terminology. That being said, I have written a query that searches within all 1800 tables in the database and will return specific tables if the name of the column that I pass into the query exists:



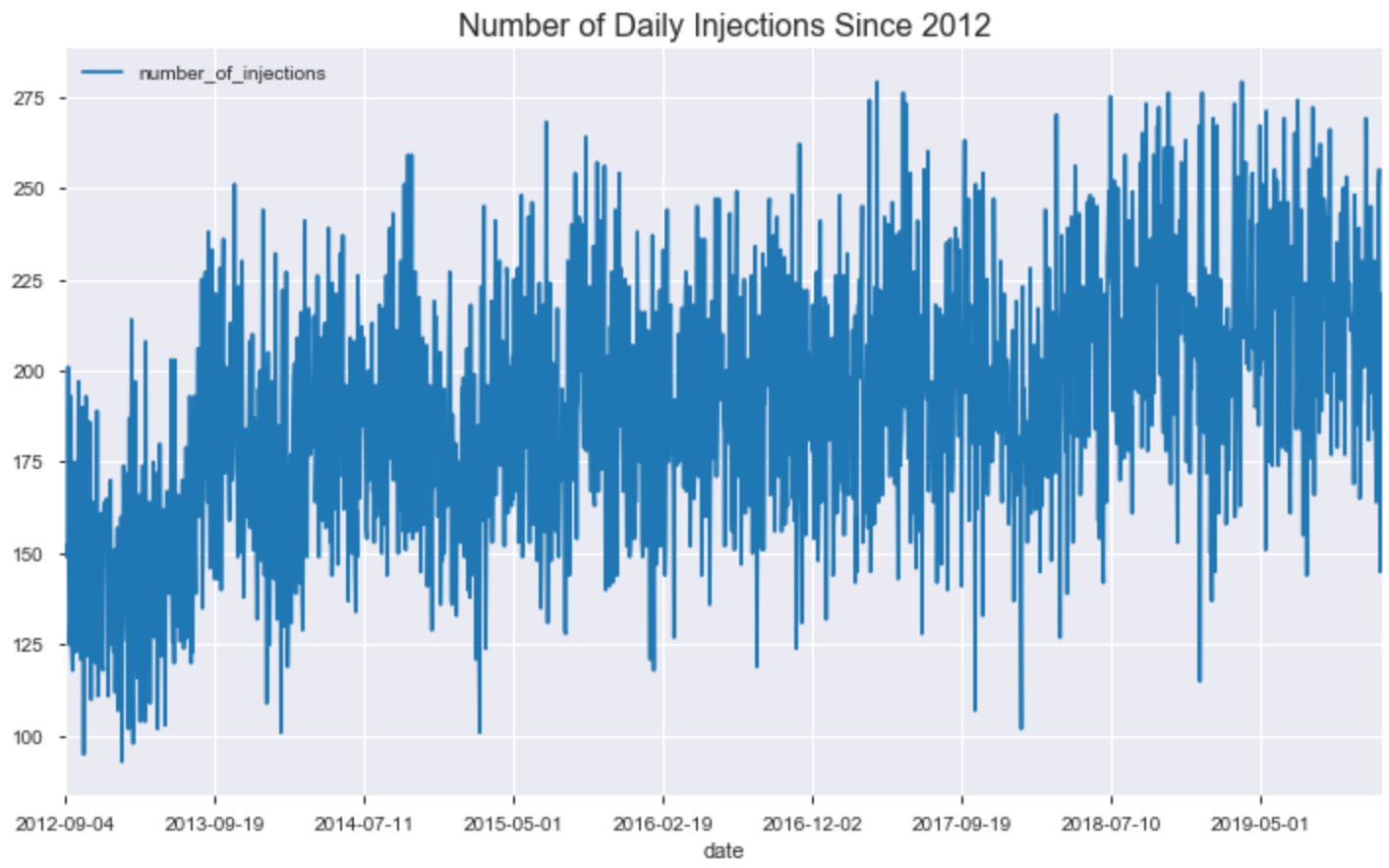
This query was incredibly helpful in allowing me to quickly locate tables that contained columns that may point to the source of the data that I wanted to use for my model. I knew, from my previous forays into the database, that the “visit type” was a column that showed codes for each patient’s visit, and after I found the corresponding descriptions, again using the query above, I was able to join these tables together and get a daily aggregated count of the number of injections for the clinic I was looking for. In addition, since I will also be forecasting this same data for the other four locations of the business, I was able to collect those data points as well and store them for future use. The query below allowed me to see data for the clinic that I was interested in obtaining results from:



Breaking down the query here, I was looking for information that was aggregated by day, from the beginning of when the clinic started funneling data into the database, which was in early 2012. I was interested in finding the actual date, the day of the week that was (since I was going to have to cut out weekends from the dataset for future forecasting, and any weekends left in the database were definitely mistakes), and the clinic location. Once I wrote this query, all I had to do was export the results to a csv file and import them into SQL. Not only did I now have the data I needed for my forecasts but I also had a way to efficiently search the database for future projects.

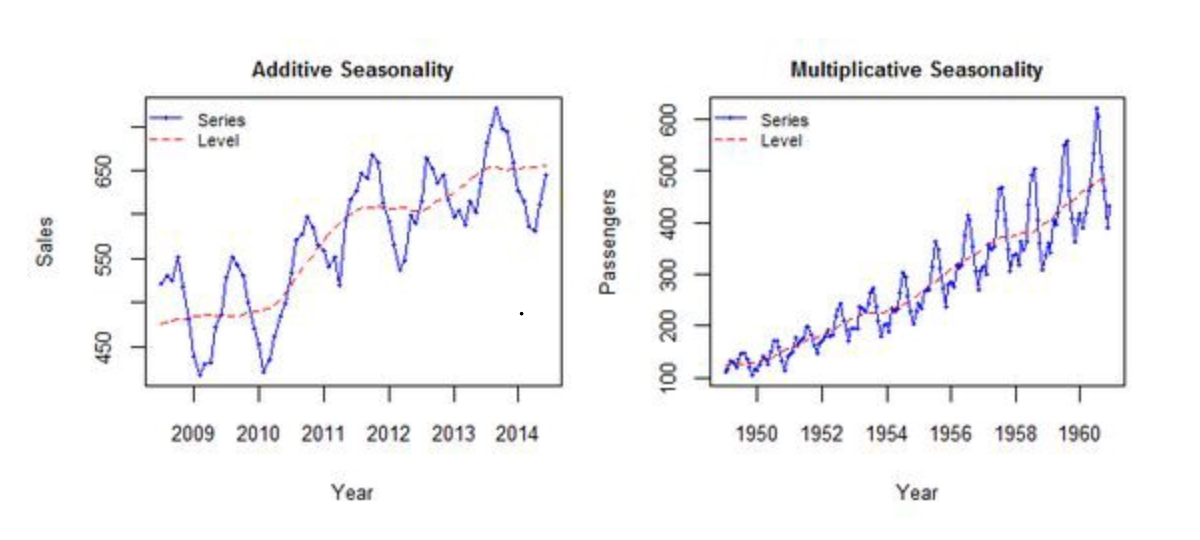
# Storytelling and Statistical Analysis

Upon opening the dataset, I wanted to get a visual idea of what the data looked like, which in my experience with time series data can be done by creating a lineplot. Using a preliminary lineplot, maybe I could begin constructing a story about the company and its rate of injections.



Great! Now I can see exactly how the injection counts have changed over the years. A good point to note here is how this plot allows you to notice points that fall outside of the normal range of injection counts, or “outliers”. You can clearly see a few points that fall close to the 100 range and the 275 range that are well outside of what the clinic normally sees during a typical day and I could consider removing later on.

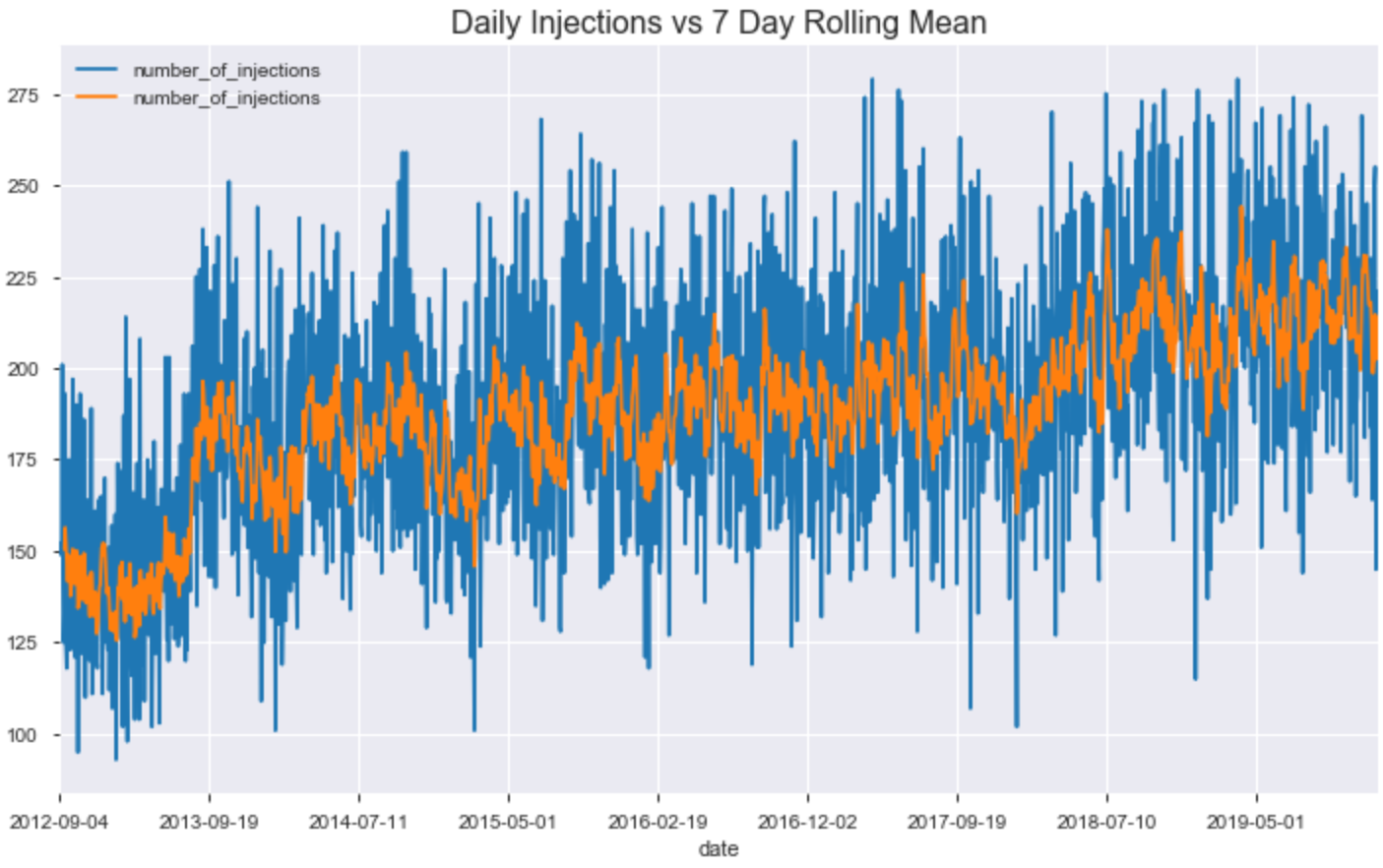
Now, ideally, I wanted there to be a clearer trend in the data. When forecasting time series data, its all about how you can pick up the trends in growth or decay. In many cases, it breaks down to the data either being “multiplicative” or “additive”. Choosing this trend type is important because we will be specifying this in our forecasting model. I think it helps to see an example of both of these so you can wrap your head around what they are.



Now let me explain what these plots mean. The additive seasonality is on the left, and it reflects a trend in the data where the magnitude or in this case (just for example) the number of injections would not really change in step with time. The values in the data do not grow or shrink the longer time goes on. With respect to multiplicative models, the magnitude of the data grows in relation with time. You can see in the plot on the right that the values in the data grow as time goes on, showing a clear seasonality trend in growth.

Comparing these charts with the line plot I created above, it was actually fairly difficult for me to parse out which type of model this should be. I could see points in the data where if you squinted and turned your head just right it could be either one of these. So, I needed to plot the data in more creative ways to try and see if I could make a definitive determination.

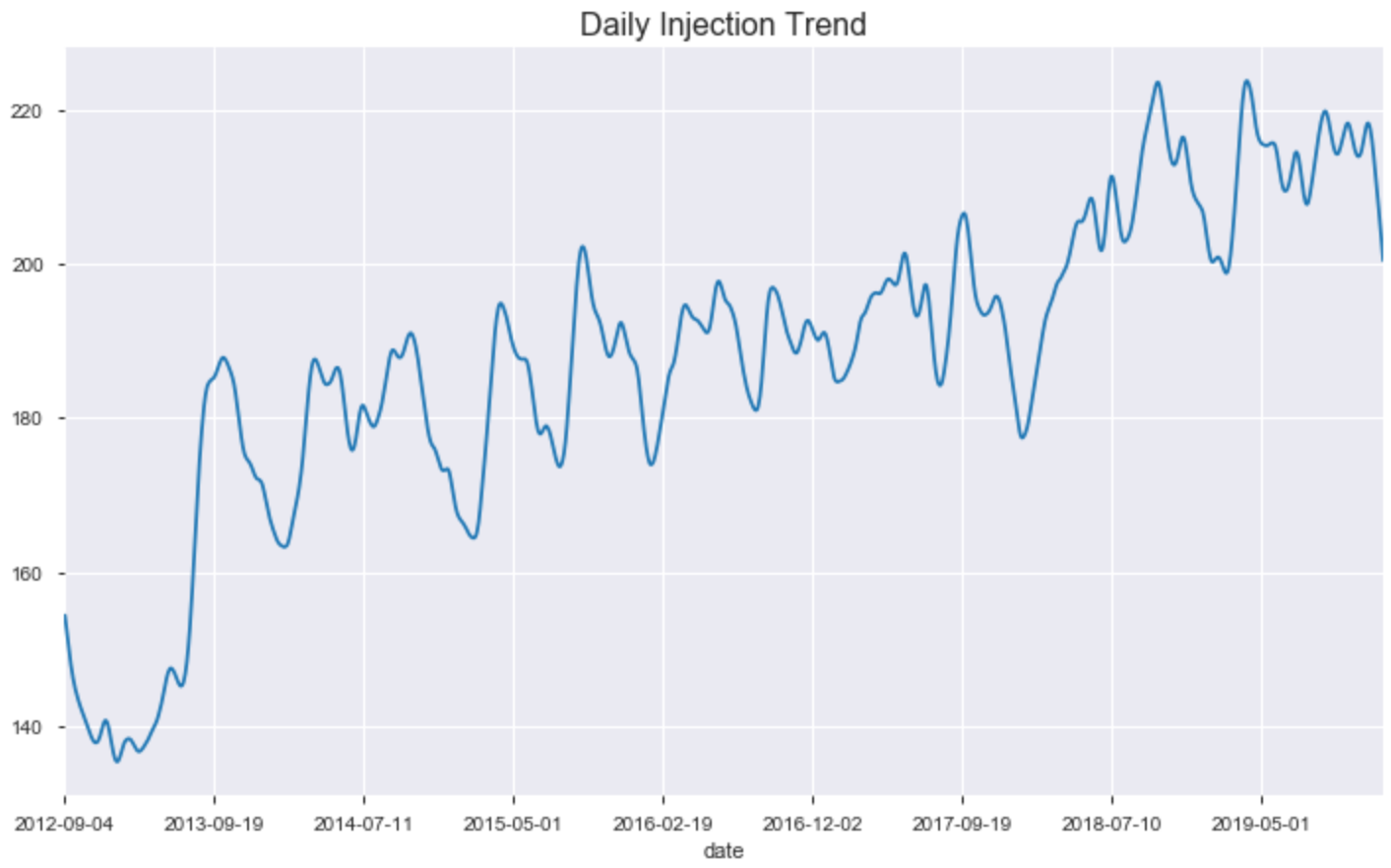
The next step I took was looking at what is called a “rolling” mean. Plotting these can be incredibly helpful in both exploratory data analysis, finding trends in your data, in addition to modeling. You can actually use rolling means to run regressions and make forecasts as well. I simply wanted to see if a rolling mean could tell me anything more about whether my data was additive or multiplicative, and also see if any granular seasonal details jumped out at me. I chose to use a seven day rolling mean, which was fairly arbitrary on my part, but was also intentional because I felt like seeing the data broken down on a weekly level may offer me better insights and cut out some of the noise.



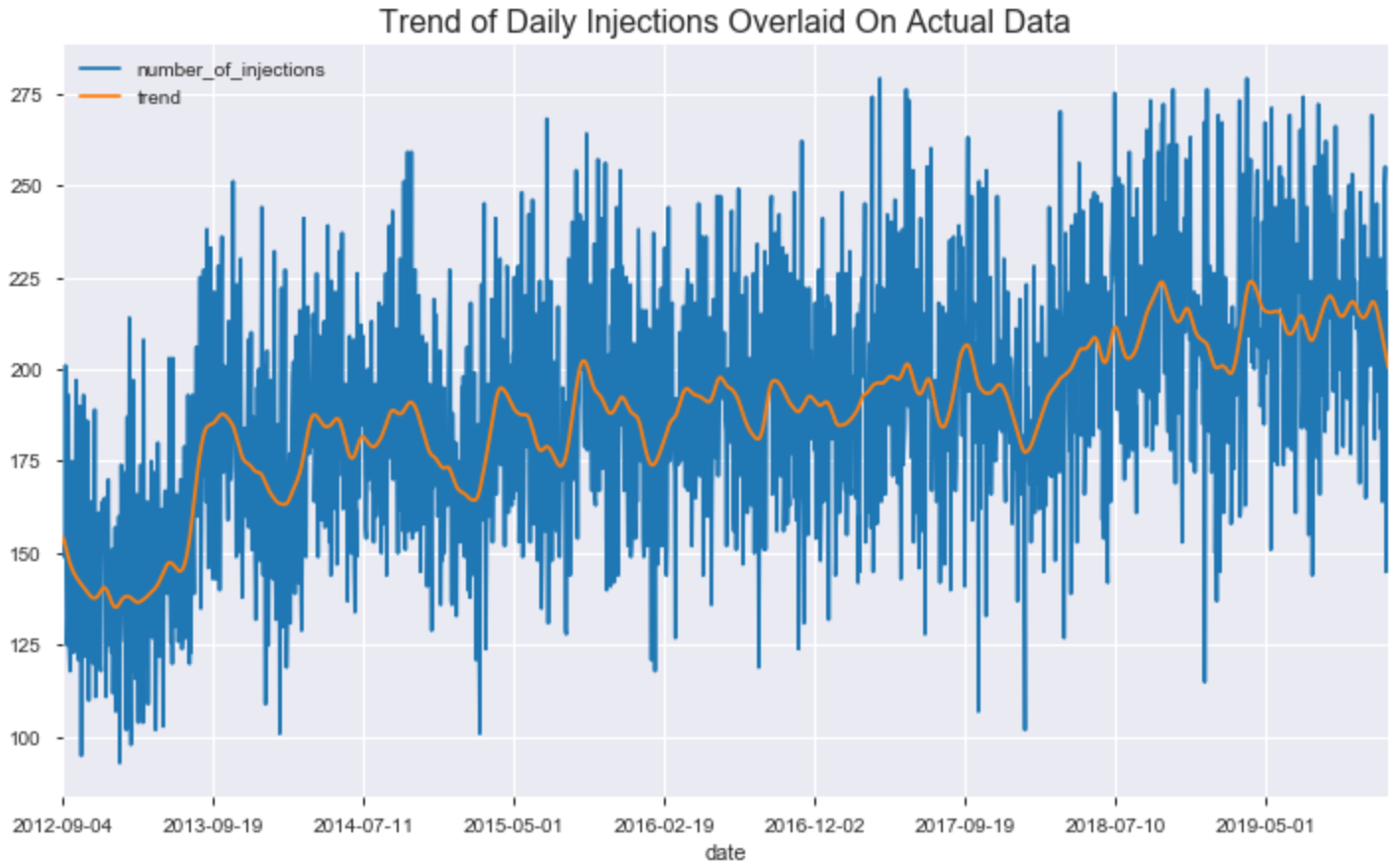
Now, this plot had not quite offered me the clarity that I was hoping for, but it did reveal some better detail in seeing how I should look at the data. Its hard to tell if the magnitude of the data is growing with respect to the time, since it could be increasing by a few injections that would make it harder to pick up. If you look at the data starting around late 2017, I was seeing a wider distribution of the points, and it felt like the data was becoming more pronounced in terms of widening with time, which made me consider using a multiplicative model over an additive one. Now that I had a better idea of what type of model I was going to implement, I wanted to begin looking at specific trend changepoints in the data that I could insert into the model.

Plotting seasonal trends is incredibly important with time series forecasting, particularly if you are going to approach the problem from an autoregression with an integrated moving average (ARIMA) standpoint. If you find seasonality in your data, then you can tune your model to the trend appropriately. I decided to plot the trends in the data a couple of different ways so that I could be confident in how I was looking at the data, and to display some alternatives to traditional seasonal decomposition.

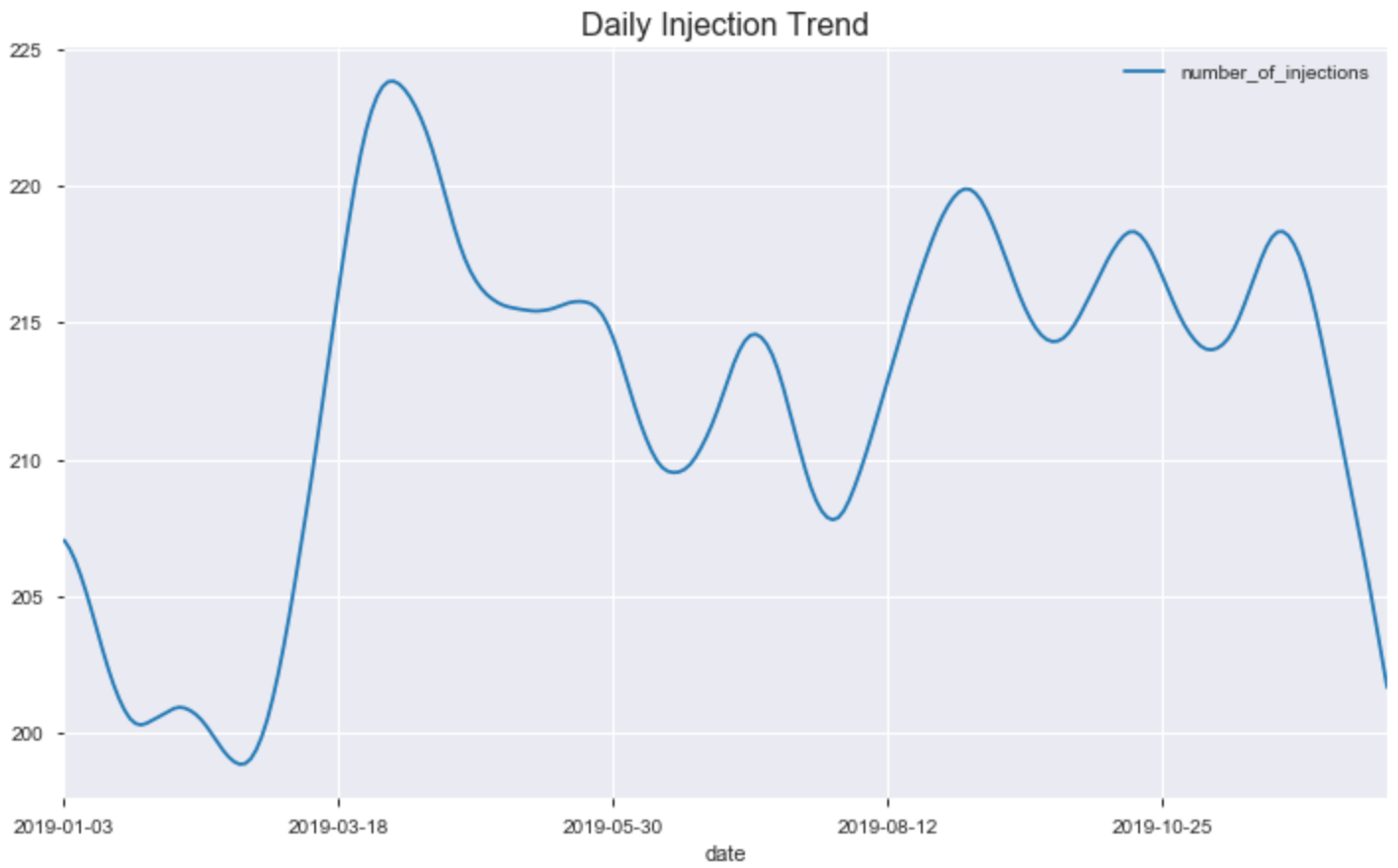
First, I used a Hodrick-Prescott filter to find a trend.



And after finding the trend alone, I overlaid it with the complete data set.

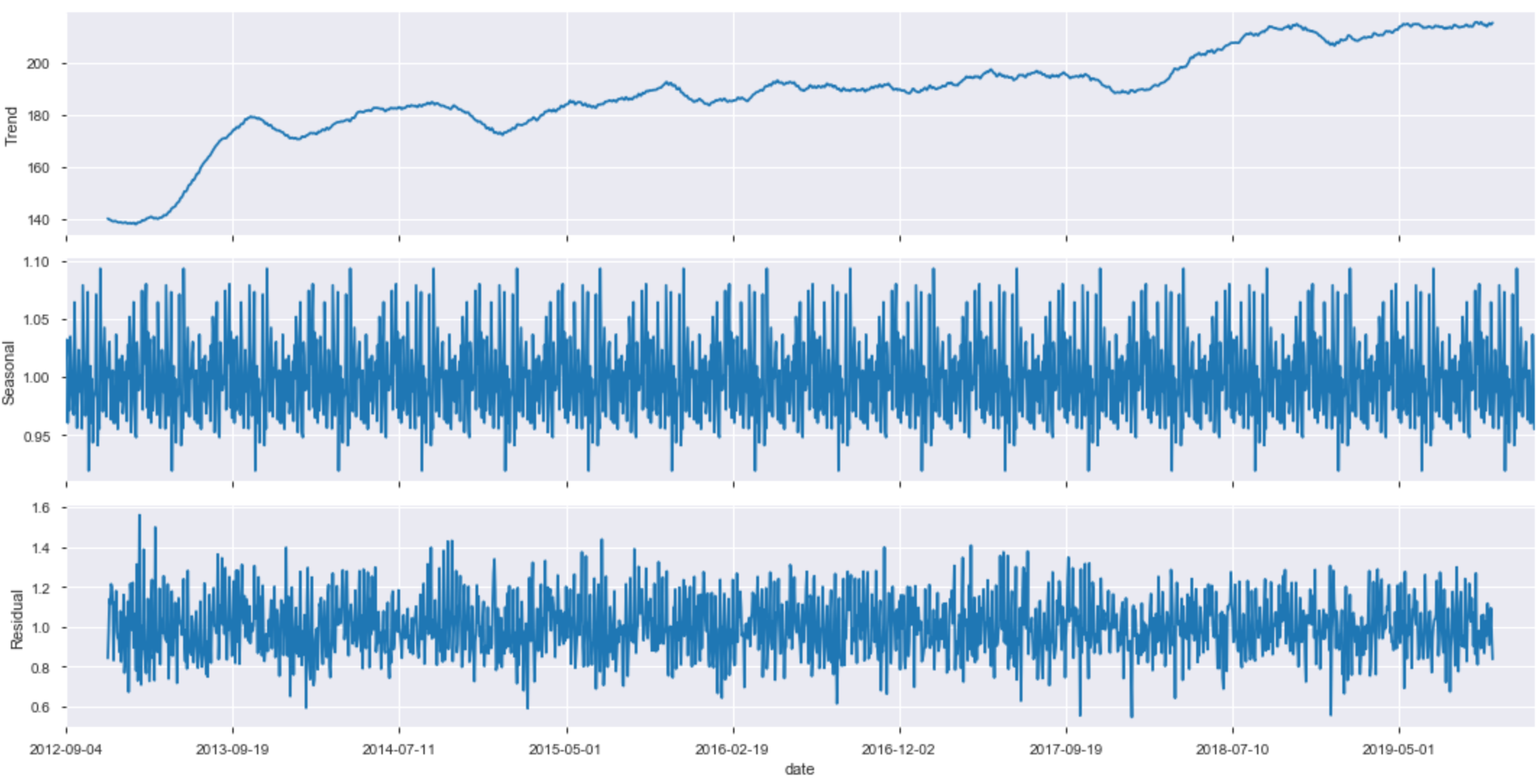


Now this was offering a look into how the data was fluctuating in terms of seasonal changes. Seasonal changes can actually refer to hourly, daily, weekly, monthly, yearly seasonality as we will see later on, but this plot was showing the general trend in the seasonality of the data as it related to the monthly level. There are some important things that we can pick up from this plot, namely that there seems to be a clear pattern in the annual data, with peaks and valleys occurring in similar places throughout the plot. Also, there are aspects of this plot that bolster the argument for the multiplicative model as well. I was curious about looking at the seasonality for the past year, to see if I could weave a more detailed story, so I plotted that as well.



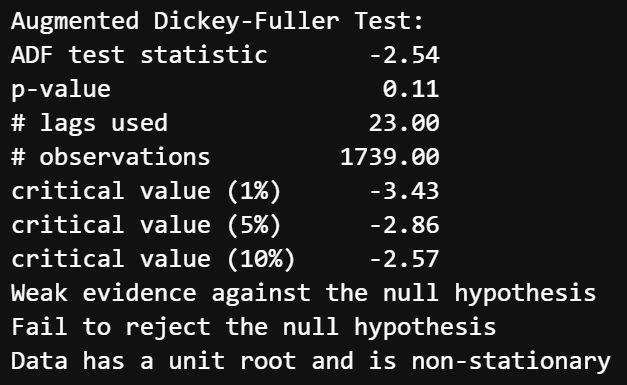
Now this shows some very interesting data points. There is a clear peak in the spring, which corresponds to higher volumes of allergy shots, with a decline into the summer, then peaking again in July and august, before making one final peak in September before falling sharply during the winter. We can see from this plot how people tend to be medicated in terms of allergy/sinus shots throughout the year. As the pollin levels increase in the spring, there is a larger amount of people with allergies. In the summer there are a couple more peaks, and again in the beginning of the fall people are beginning to get a little sick and are coming in for more shots related to colds perhaps. When I showed this to the CEO of the company, he was very excited to see how we can begin to stock our supplies of medication during these times so that we don’t allow any of our supply to expire. I would plot this data again later on in the modeling stage using Facebook’s Prophet algorithm, which showed an annual trend for the data since 2012 that was also very helpful.

Another way of looking at trend and seasonality is through using ETS (error, trend, seasonality) decomposition. Using this helped to cement any assumptions I was making about the data.



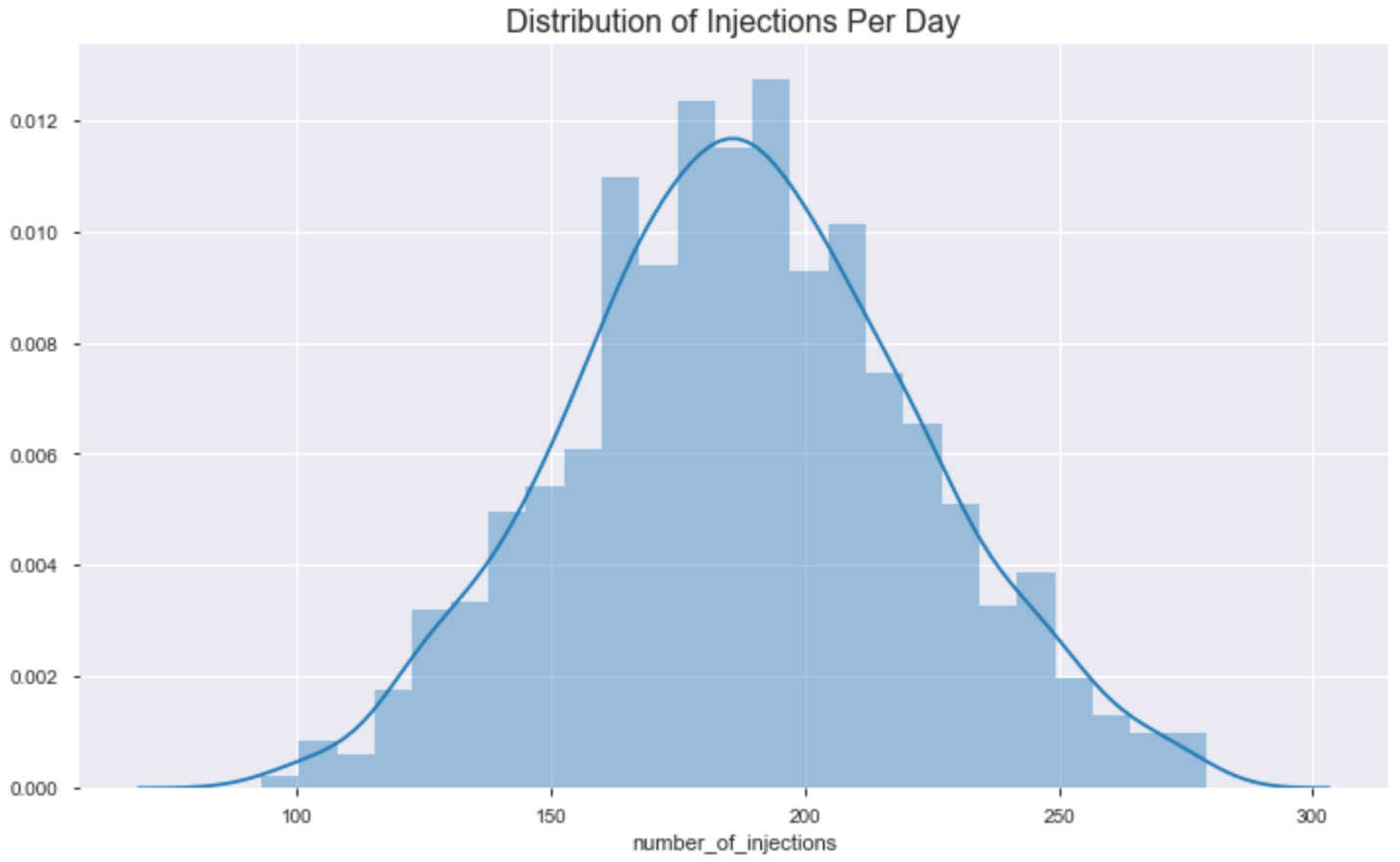
Here we can clearly see that there are obvious seasonal trends to the data, and the upward trend is exhibited in the top portion of the plot. Seeing this breakdown made me feel more confident in the seasonal inferences I was making.

My next course of action, now that I have plotted the data several ways to interpret the seasonality, was to definitively determine whether my data set was stationary or not. Stationarity refers to data that does not follow any seasonal trend, and while we seem to have confirmed that, it is always nice to have a statistical test to back your assertions up in the boardroom. That being said, I ran a simple Augmented Dickey-Fuller test to see if my p-value was significant for the data, indicating stationarity.

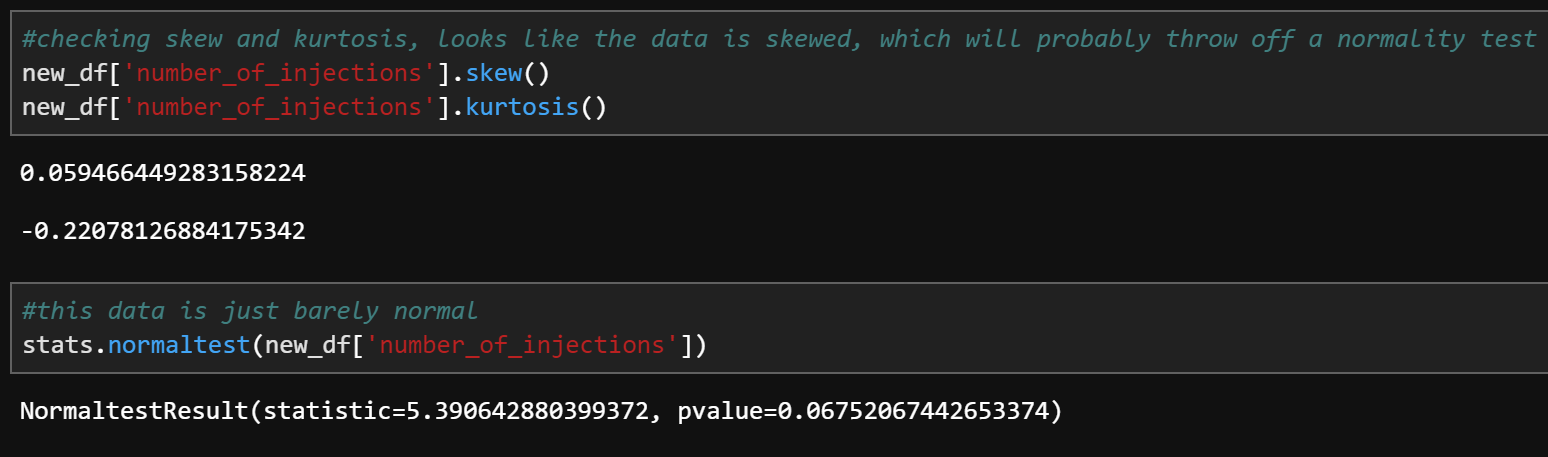


Now I was finally certain of the seasonality of the data in terms of a statistical test, and I could also plot a course of action if I chose to follow the ARIMA path later on, which would involve making my data non-stationary (Note: I tried several AR, ARIMA, ARMA, SARIMA models later on in this project and the were making predictions that had higher residual error than other models, so I abandoned them).

Now I also needed to look at the distribution of the injection counts and consider transformations if anything seemed too skewed or kurtotic. Since this time series only contains one column of data, the statistical analysis was relatively straightforward in terms of looking at the distribution. I desperately wanted the data to be relatively normally distributed since regression models deal better with data that is in that form and transforming the data would involve making the interpretation of the predictions less straightforward.



This plot, thankfully, looks fairly normal, especially with the curve superimposed. Now looking at the skew and kurtosis, in addition to performing a Shapiro-Wilk test for normality would help me confirm that my data was normally distributed.



Now values of less than -1 or greater than 1 are considered “skewed”, with values higher than 3 considered to be highly kurtotic. Luckily, we are not seeing either of these thresholds exceeded with this data. Furthermore, the normality test has come up insignificant (p-value less than .05), which means that our data is normally distributed. After running these test, I felt as though the next step was to begin modeling and tweaking hyperparameters in the various models to find the one that would produce the highest-quality predictions.