Paycheck Protection Program Event Study Analysis: The Process

Brian Keller

Step 1: Data Manipulation

The first roughly 100 lines of code are solely for manipulation and formatting of the initial three data sets. From the CRSP data, I imported only the “date”, “permno”, and “ret” columns so as to eliminate as much excess data as possible and simplify the rest of the process. I took all of the Fama-French data and all of the PPP data for only the companies that had a date in the column “ppp_sec_filing_date1”.

After importing the data, a lot of time was spent cleaning the data and merging the CRSP data with the market return column in the Fama-French data. I made a list with all of the permnos for the companies remaining in the PPP data set. This will be used later in the process.

The last thing done in this section of code is cutting off the first 200 days of data. Since I am not using the exact dates from the PPP loans during calculation, I decided to keep roughly four months of data that contains all of the PPP loans. This also significantly reduces the amount of time it takes to run the code. One thing to note is that even though there were roughly 300 companies which had received loans according to the PPP data set, I only had CRSP data for 249 of those companies.

Step 2: Defining an Event

As stated before, I didn’t use the exact dates from the PPP data as events, instead I took a chunk of data surrounding all possible PPP loan dates and found events in it. The reason for this is because the CRSP companies that did not receive loans don’t have event dates tied to them, so I would have to create my own events according to a formula such as the one actually used in this code. If I do this for the 3000 non-PPP loan receiving companies, it makes sense to me to also create events for the 300 PPP loan receiving companies, as it is not exactly a guarantee that the PPP loan resulted in a large change in stock price on a given day.

I found most of this code online when looking for how to perform finance event studies in Python. It generally follows the McKinlay event study analysis techniques so it should be applicable in this situation. The basic idea is that the user defines an “event” when the difference between the market and individual stock return on any given day exceeds a certain threshold. In this case, I chose 0.03, or 3%. This means if a stock performs 3% better or worse than the market on a given day, an event is marked on that date for that company.

The events are stored in a pandas data frame as either a 1 for positive events or a -1 for negative events. This will be important later.

Step 3: Calculations

Now that we know when events took place, it is time to calculate the abnormal return and cumulative abnormal return. This is done in a series of nested for loops which makes it wildly

inefficient and takes a very long time to run. The calculation process happens twice: once for positive events and once for negative events. The process is identical for both.

The first two for loops find the index of the event in the data frame and gather a specified window of data before and after the event. In this case, the window is set to 20 days, as recommended by the initial author of the code. It stores the 41 days of data in a dictionary and moves on to the next chunk of code. This code calculates the beta, alpha, standard error, and abnormal return for each event period. The abnormal return is stored in another dictionary.

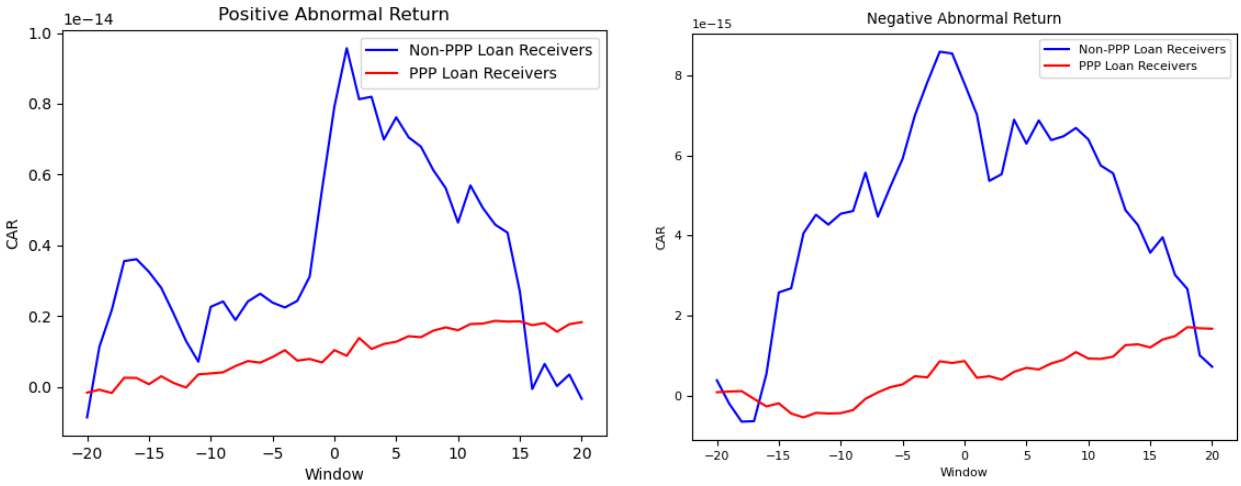
Once all calculations have been done for the 3000 non-PPP loan receiving companies, the data is stored in a new data frame and manipulated to prepare it for plotting. Also, the cumulative abnormal return is calculated at this point through a simple cumulative summation function.

As I said, this was just for the non-PPP loan receiving companies. Now I repeated this entire process, but for the 300 PPP loan receiving companies. Using the same trigger difference, and the same time window, events are found and an event period of 41 days' worth of data is stored in a dictionary. As before, each event gets its own dictionary, has its beta, alpha, standard error, and abnormal return calculated and stored in a separate dictionary. Again, once all the companies have been completed, the data is put in a data frame and manipulated to prepare it for plotting. The cumulative abnormal return is calculated in the dame way as before. (Quick side-note, there is a separate progress bar for the PPP-loan receiving companies to track that progress as well)

Step 4: Plotting

The plotting process is very simple. It uses a Python module called matplotlib. I fed the data into a few premade functions and saved the results. I plotted all positive events from non-PPP loan receiving companies alongside the positive events from PPP-loan receiving companies. Similarly, I plotted the negative events for both groups alongside each other.

The x-axis on the graphs is titled "Window" and shows 20 days before and after the event, with "0" being the event itself. The y-axis is titled "CAR" and is the cumulative abnormal return. These are the results:



Step 5: Analysis

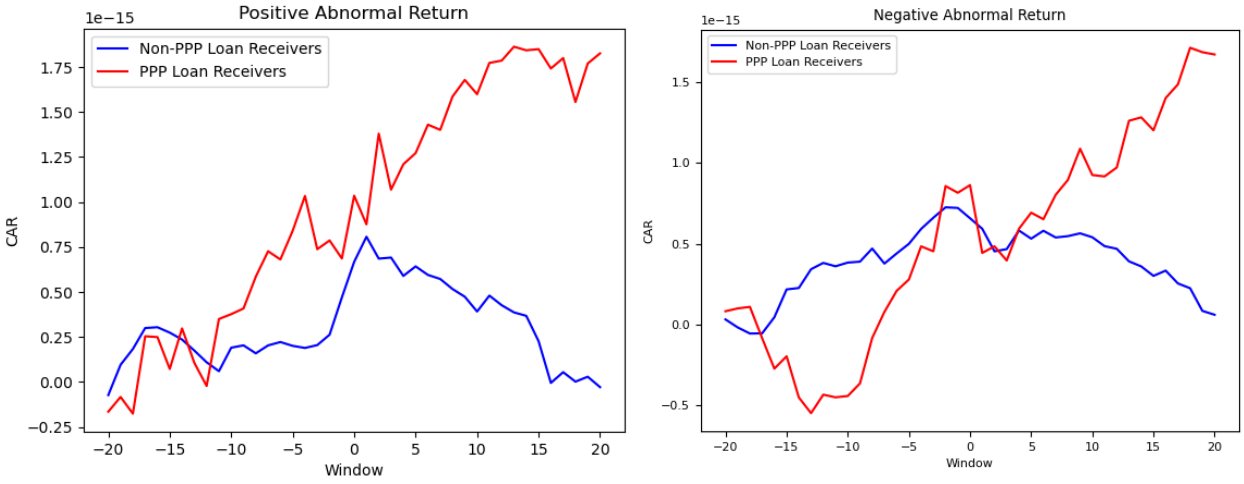
These graphics obviously leave a bit to be desired in terms of explanation since the PPP loans seem to have a negative affect on abnormal returns. The companies which received the loans have events with much less volatility surrounding them, while the non-PPP loan receivers seem to have much larger fluctuations (more abnormal returns) both leading up to and following an "event". Does this mean the government gave PPP loans to more stable companies? Or does it mean the event of receiving the loan did not give investors enough reasoning to increase the traded volume of this company surrounding the loan date. My first thought is that being given a loan should be a sign of either weakness in the company or promise that the government sees in it. Either way, it should be enough of a reason for investors to flood the company with more volume than usual leading to abnormal returns, but this isn't the case.

One interesting thing to note is that the non-PPP loan receiving companies have vastly greater abnormal returns likely meaning the betas of these companies are significantly greater than the betas of the PPP loan receiving companies. Maybe this exact statistic was used when selecting which companies to give loans to. In some way, a wildly volatile company may be too weak to survive a large downturn in revenues, even if it receives a loan from the government.

Whatever the reasoning, it seems the recipients of PPP loans were generally more stable companies in the past four months.

Step 6: Potential Errors

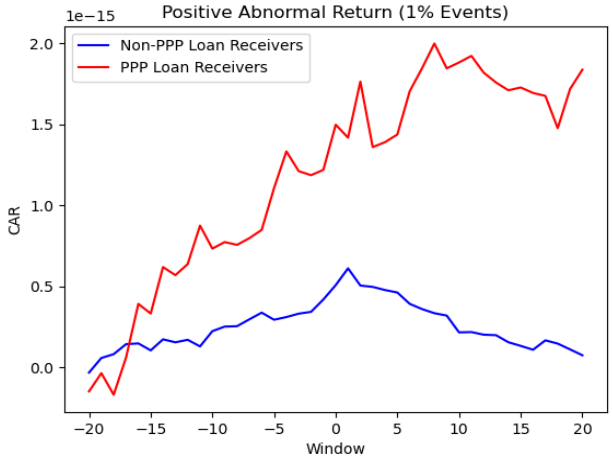
The first potential error I thought of is with the CAR. Since I'm not entirely sure what I'm doing when calculating this, I decided to divide the CAR of the non-PPP loan receiving companies by 11.86 which is 2954/249(the number of non-PPP loan receivers divided by the number of PPP loan receivers). This gave me very interesting results which make MUCH more sense to me, even if dividing by 11.86 makes no sense. These are the results:

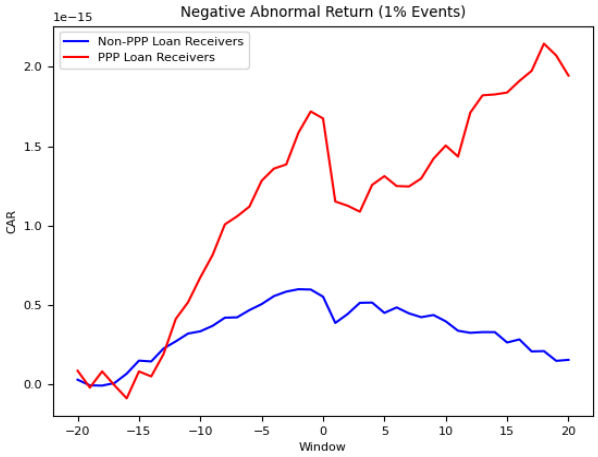


Since this seemed to make more sense, I decided to keep this in my code when running the rest of my tests of potential errors. We will call these the "adjusted" results.

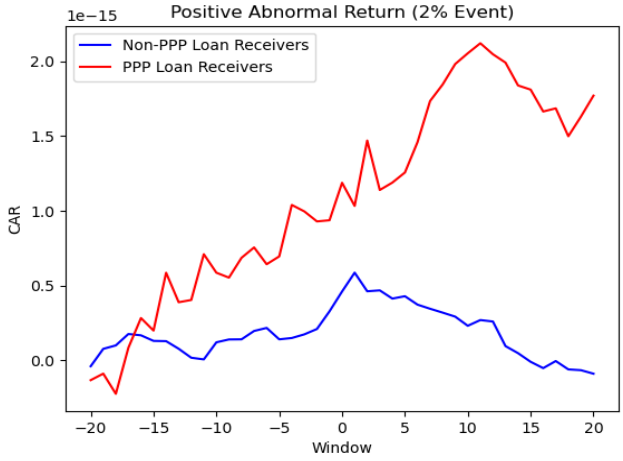
The next biggest potential error in my mind is the classification of an event. I used 3% as my event trigger but for all I know, the author of the code I found could have pulled that number out of thin air. With that in mind, I decided to test a few more triggers: 1%, 2%, and 5%. The results are as follows:

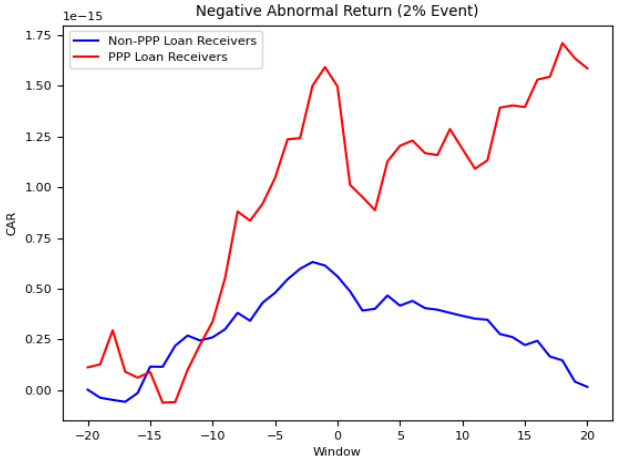
1 Percent



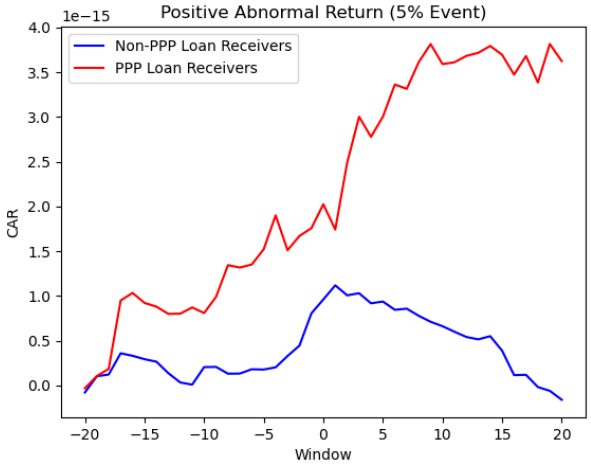


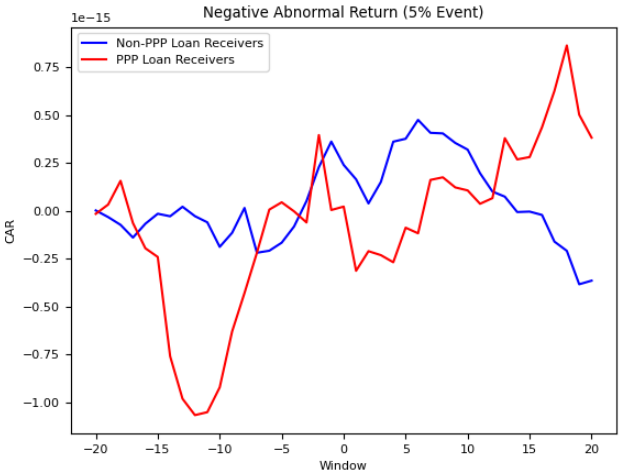
2 Percent





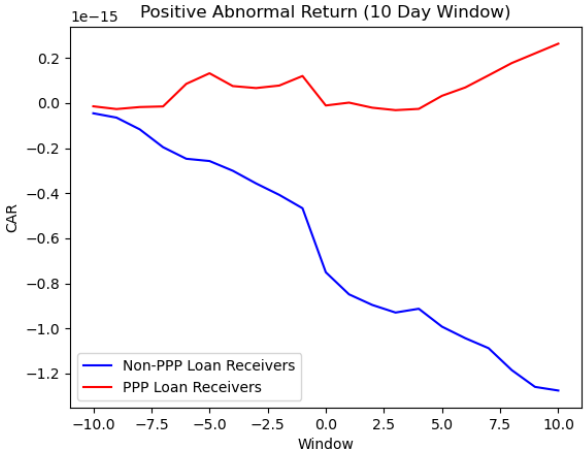
5 Percent

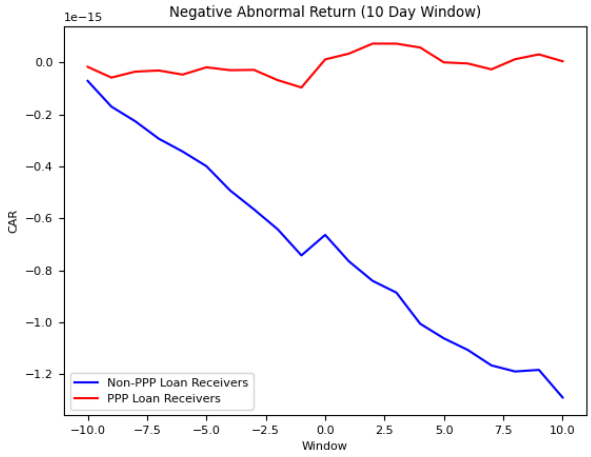




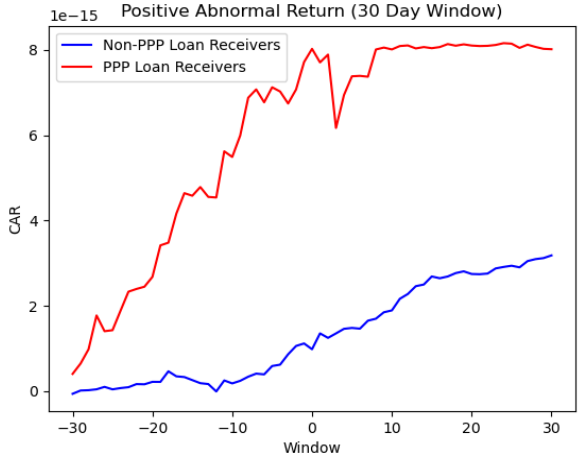
The next potential error is the window period. Again, I tested a few more options (all with 3% events): 10 days, 30 days and 45 days. For the 45-day window, I expanded the data in the beginning from roughly four months' worth to roughly six months' worth. The results are as follows:

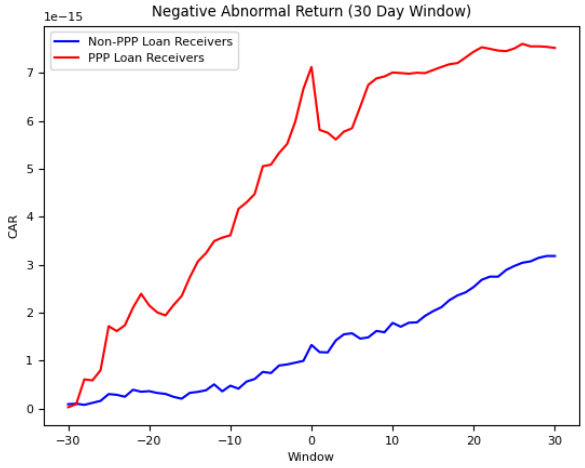
10 Day Window



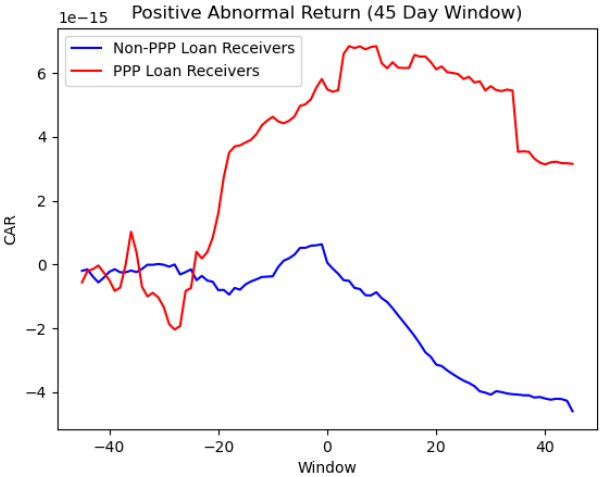


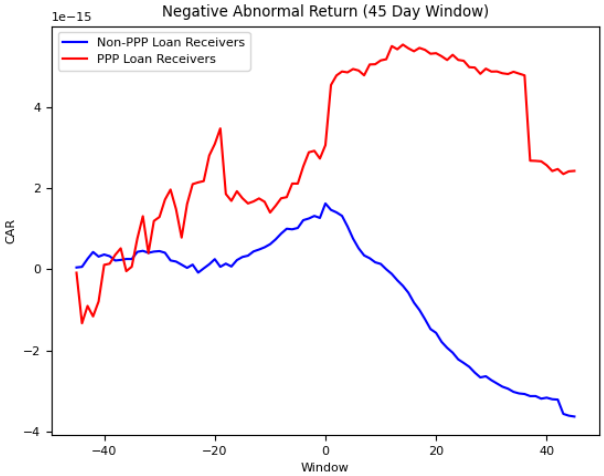
30 Day Window





45 Day Window





Step 7: Conclusion

This entire process was entirely new to me, and I've never worked with data on such a scale before. All this means is that I'm sure there are plenty of places I may have taken a wrong turn. Also, since much of the actual calculation process was done with code that wasn't authored by me, I'm not entirely sure if it follows the process I would've used. Other than that, the biggest flaw in this code is efficiency. Because of the nested for loops, it takes a significant amount of time to run for large data sets. If I started from scratch, I would likely use the pandas .apply() feature to perform the calculations in much quicker time.

As for the results, I believe the 20-day window combined with the 2% event trigger, along with being adjusted by the proportion of non-PPP loan receiving companies to PPP loan receiving companies provides the best results for interpretation.

In that case, it seems to show that the abnormal returns surrounding the events for PPP loan receiving companies are significantly greater than those that didn't, which is what I would expect to happen when news of this significance is released.

Sources:

This is the main source for the calculations portion of the code:

http://esocialtrader.com/event-studies/

The entire Python program can be found on my GitHub at this link:

https://github.com/blkeller92/PPPEventStudyAnalysis