Treasury Inflation PCA Signal (Rough Draft)

diego[dot]alvarez[at]colorado[dot]edu

February 1, 2025

1 Introduction

This paper focuses on a series of signals that are generated based on comparing different inflation term structures to trade Treasury Futures. In this case, a signal is generated by decomposing two inflation-based term structures and examining their spread. The motivation of this paper is that two inflation term structures that track similar or the same inflation rate should trade in-line with each other and that small dislocations will occur, which can be harvested. Eventually, after testing the signals' performance, the focus will shift to examining the risk premia and the economic interpretation of the signals' performance.

Surprisingly, the signal has high performance and consistent results for various sovereign bond futures. In this case, due to data constraints, this model is applied only to US Treasury Futures.

2 Signal Generation

First begin with two inflation-based term structures in this case the Treasury Breakeven Term Structure and the Inflation Swap Term Structure are used. These two curves can be decomposed into their relevant k factors via PCA. Like most fixed income term structures, they usually require at most 3 components, which reference level, slope, and curvature.

At this time, signals and dimensionality reduction methods will be kept simple, the signal is defined as the difference between fitted values of each PC. Before fitting the PCA model, the inflation swap rates and breakeven rates are logarithmically transformed. For $i \leq 3$ component of the curve the signal s can be generated as

$$s_i = PC_{i,curve1} - PC_{i,curve2} \tag{1}$$

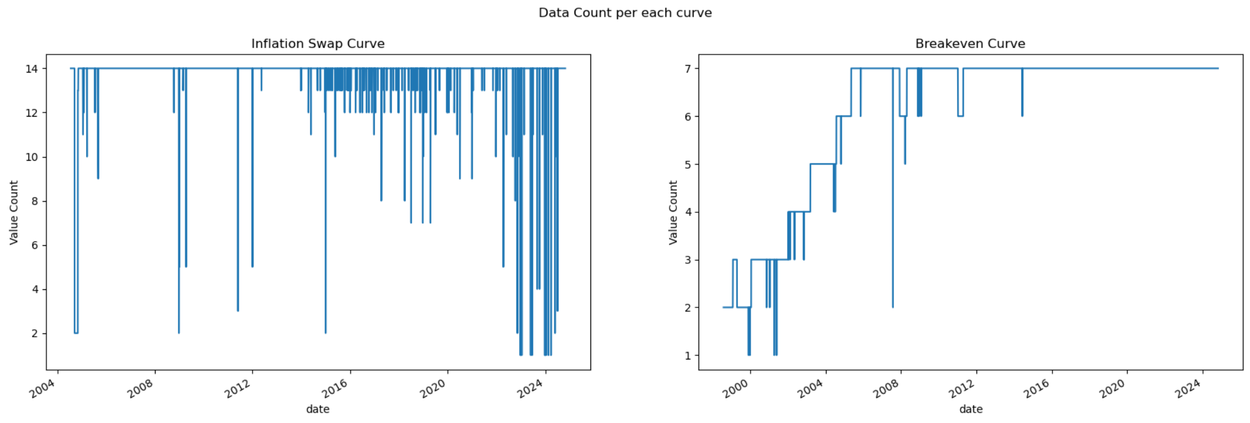
Notice how the choice of which curve isn't specified. That is because there isn't an intuitive reasoning for which PC embeds the risk premia. One could infer how the risk premia is embedded and test for it, but for the time being, one way will be set. The signals' performance will be conditioned purely on the sign of the signal without any optimization. Therefore, multiplying the signal by -1 is equivalent to shorting the signal. Later through some empirical research, it shown that it is quite easy to find *which way* to trade the signal and that its highly unlikely otherwise.

Then each signal is applied to each Treasury futures contract. All returns are expressed in basis points across each futures contract, so the results are comparable. Two portfolio optimization methods are used the first is equal weight and the second is equal volatility contribution. The equal volatility contribution is done in basis points returns which isn't completely accurate of the portfolio's performance but works in this case. Volatility is defined as 30 day exponential moving standard deviation.

2.1 Data Quality

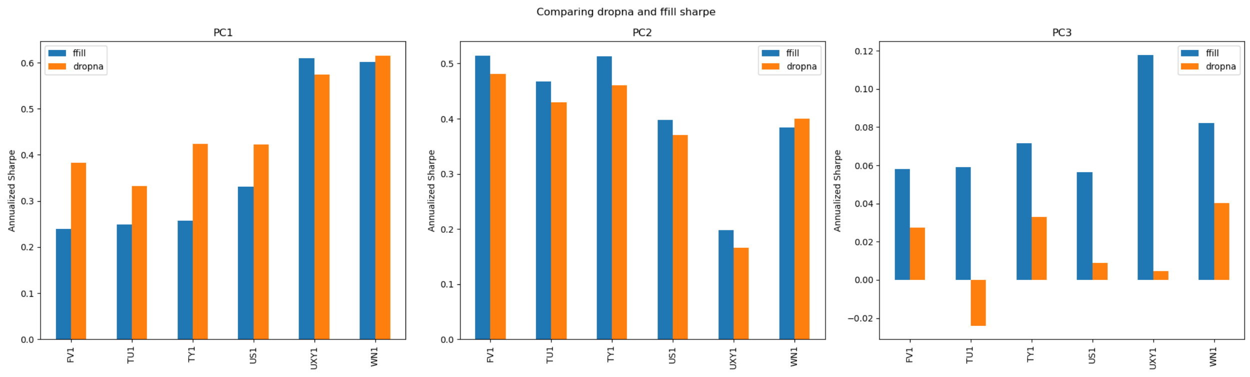
There appears to be some data quality issues with both Treasury Breakeven and Inflation Swap data, which is collected from Bloomberg Terminal. Assuming that all tenors of the curve are kept and inputted into the

PCA, and each PCA computation is done per-curve, its a requirement for all of the data to be present for that day. If a tenor is missing for a specific curve on that day the whole PCA model gets thrown out for that day.Below is plot of the missing data.



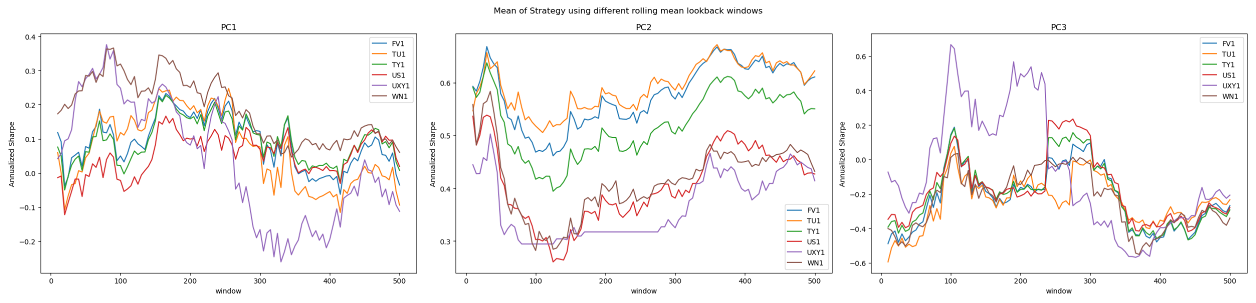
Normally with the breakeven curve, since different tenors started during different times, multiple PCA models would have to be *stitched* per each new addition. That isn't of concern here since the inflation swap data doesn't start until 2004 which is when all breakeven tenors are active.

For the most part the data cleaning will involve using simple approaches such as panda's `pd.ffill` function or using a simple rolling mean. The question to answer is how does data cleaning affect the results. All of the comparisons at this stage will focus on the sharpes of each signal. Start with the comparison of sharpes between `pd.dropna` and `pd.ffill`



The `pd.ffill` method has similar sharpes across each return but the third principal component shows the greatest difference. This is a bit intuitive since level (the first PC) isn't necessarily tenor specific and slope is really dependent on an upper and lower set of tenors, while curvature is more tenor-sensitive. The third principal component is also very sensitive to changes much since it accounts for such a low amount of explainable variance. `pd.ffill` is also a reasonable cleaning method for real world application.

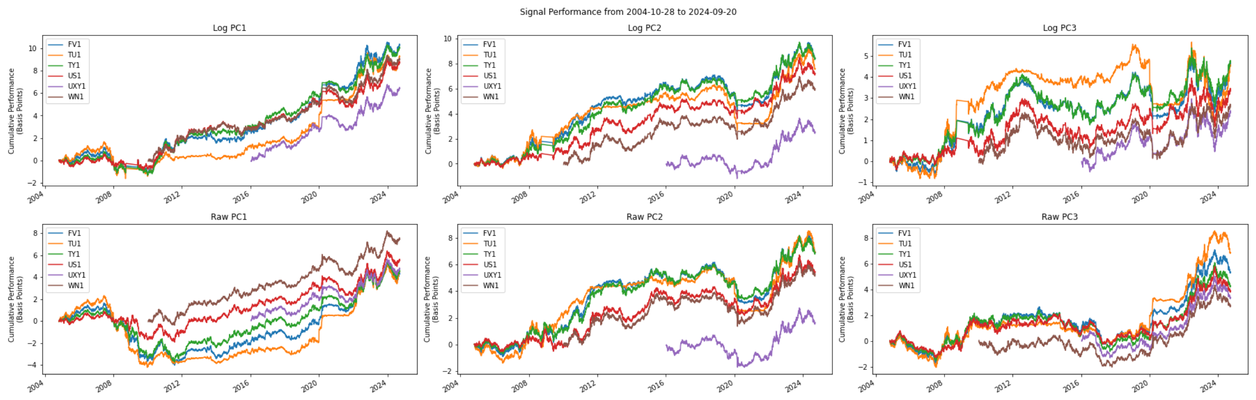
Other cleaning methods include using a rolling mean approach. In this case it isn't clear what kind of window to use. Rather than using something a little intuitive like 10d or 20d, iterate through windows and measure the sharpes. The idea is to see how *well-behaved* the sharpes are. Iterating through rolling mean window of size 0 days to 500 days incrementing by 5 days and calculating the sharpes get.



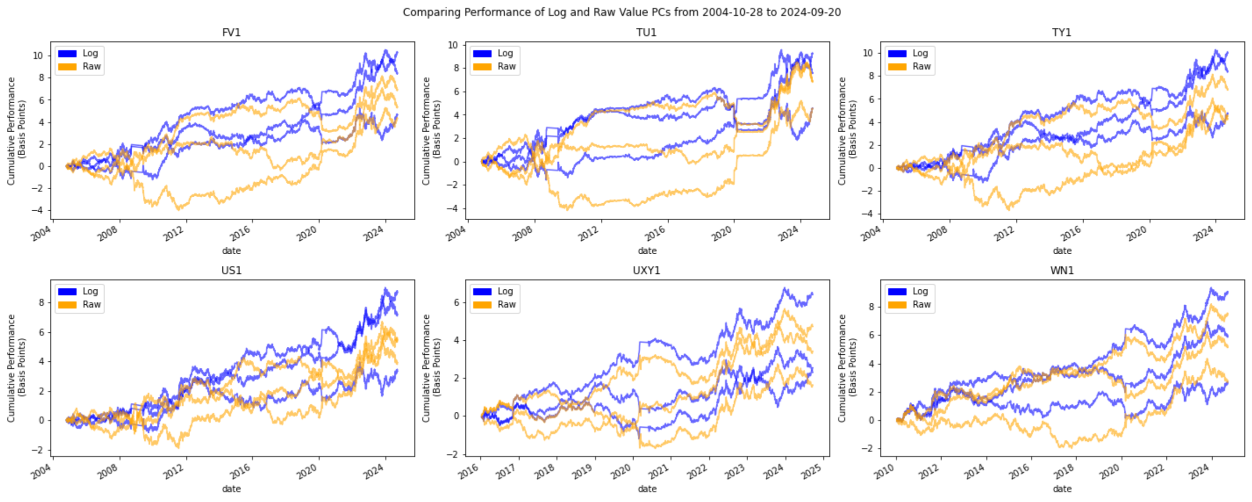
In this case it does appear that there are local structures of the sharpes. As the window changes they don't seem to taper off to a specific local sharpe other than the 2nd principal component. From this result it isn't particularly clear and `pd.ffill` appears to be the better option. This will be the baseline data inputted.

3 Initial Results

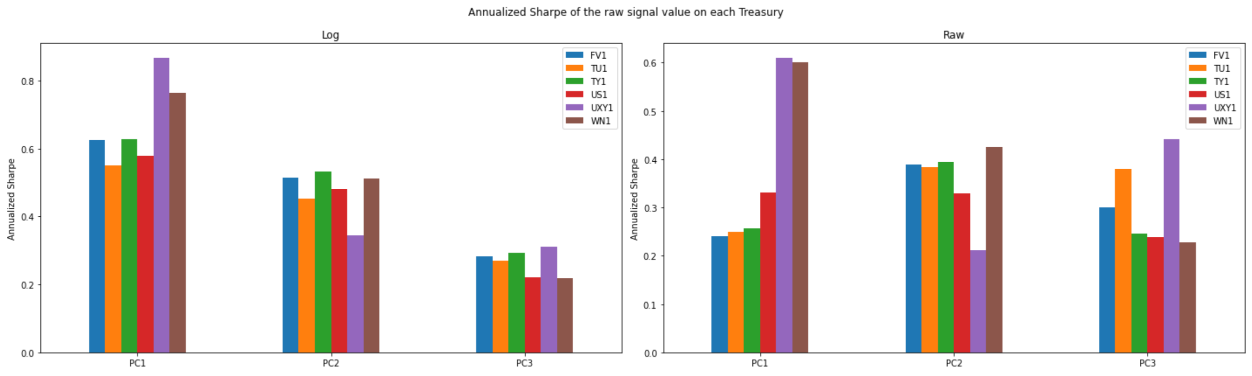
Start with applying the original signal. In this case take the difference of each principal component as the *Treasury Breakeven Curve − Inflation Swap Curve*. It appears that this signal should be shorted, for the time being let's set the signal to be short. Later on a simple model will prove that the shorting the signal is a reasonable assumption.



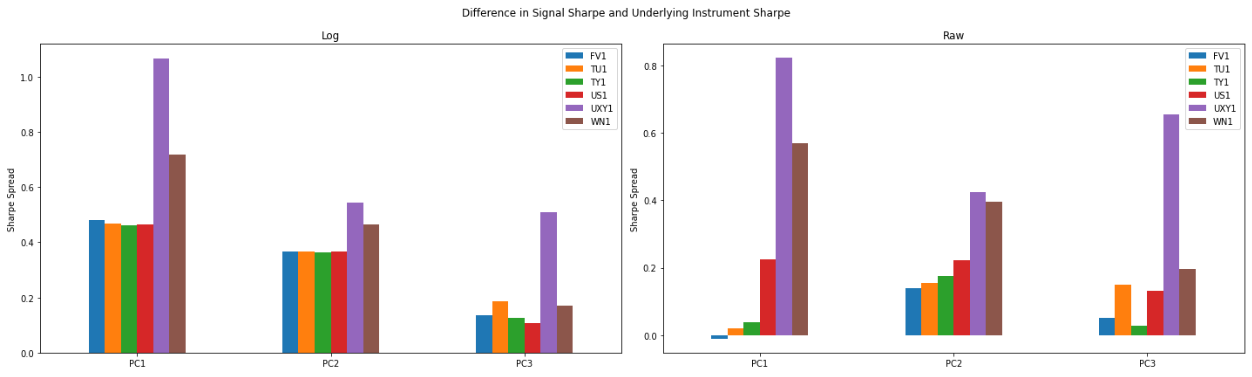
Comparing log rates vs raw rates directly



Now analyze the sharpes of each signal

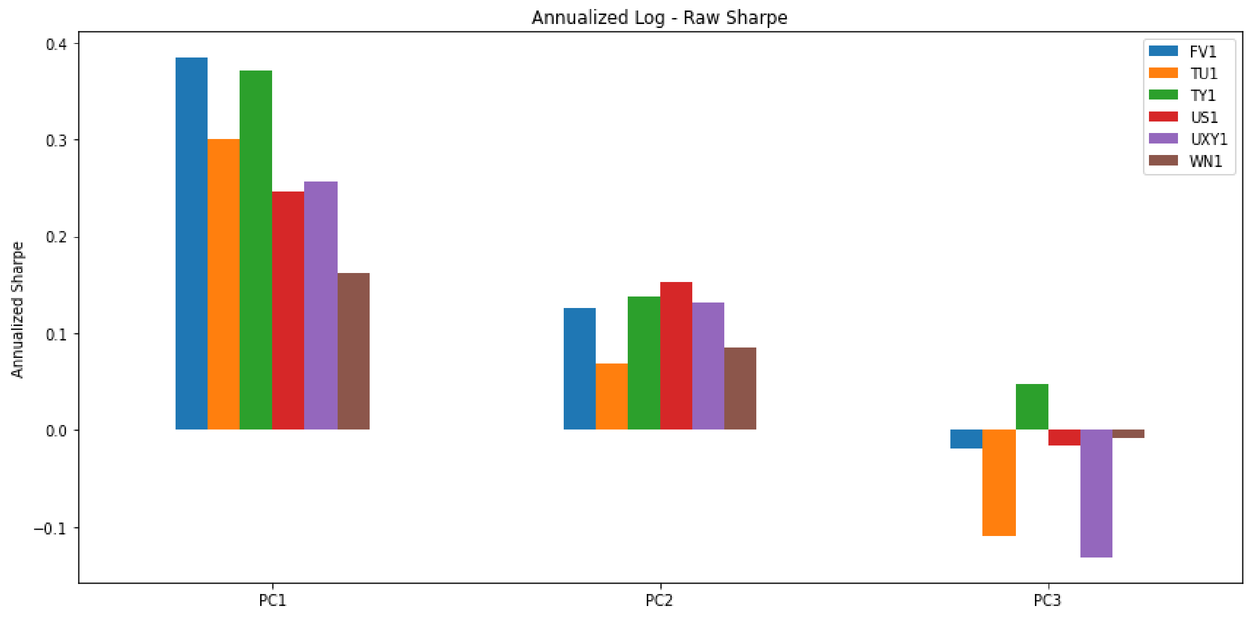


Since this is a slower moving strategy (trading daily) the sharpes need to be compared to the underlying instruments. Below is the signal sharpe after subtracting out the underlying sharpe.



In this case UXY1 (Treasury Ultras) sharpe spread is likely an effect that they were invented back in 2016 and the return's history occurred under one monetary policy regime.

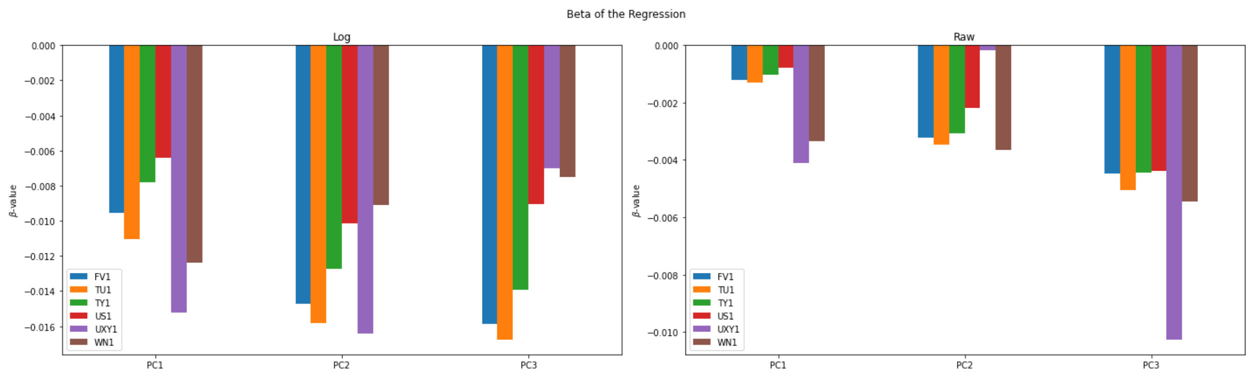
In this case its obvious that log rates vs raw rates before applying PCA is a better choice other than small underperformance of the third principal component.



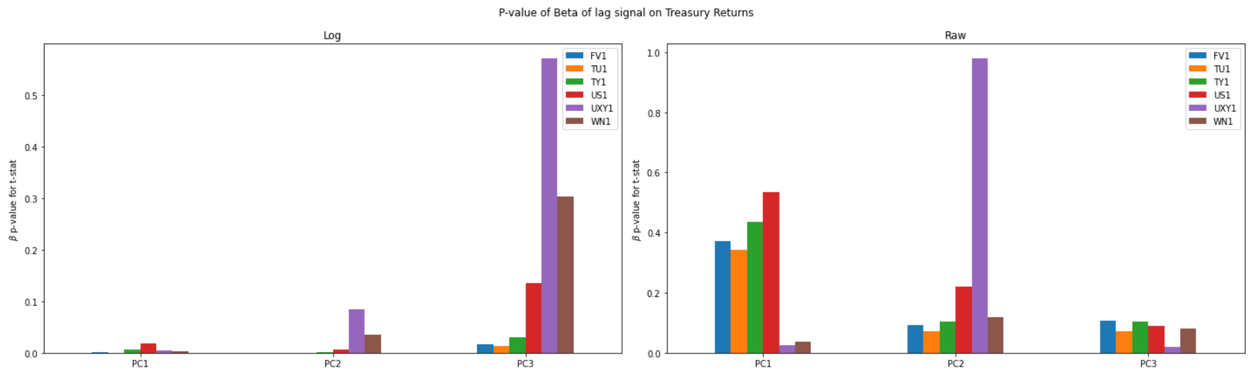
4 Signal Direction

4.1 Full Sample In-Sample OLS

In the Initial Results section the signal was assumed to be short. That was an assumption based on *guess-and-check*. Its not an adequate method in this case; start by using OLS as a way to find the *directionality* of the sign. The regression is the lagged signal against each Treasury Future's return. The plot below shows the β of the regression which implies that the signals should be shorted. Log rates have even greater absolute βs.



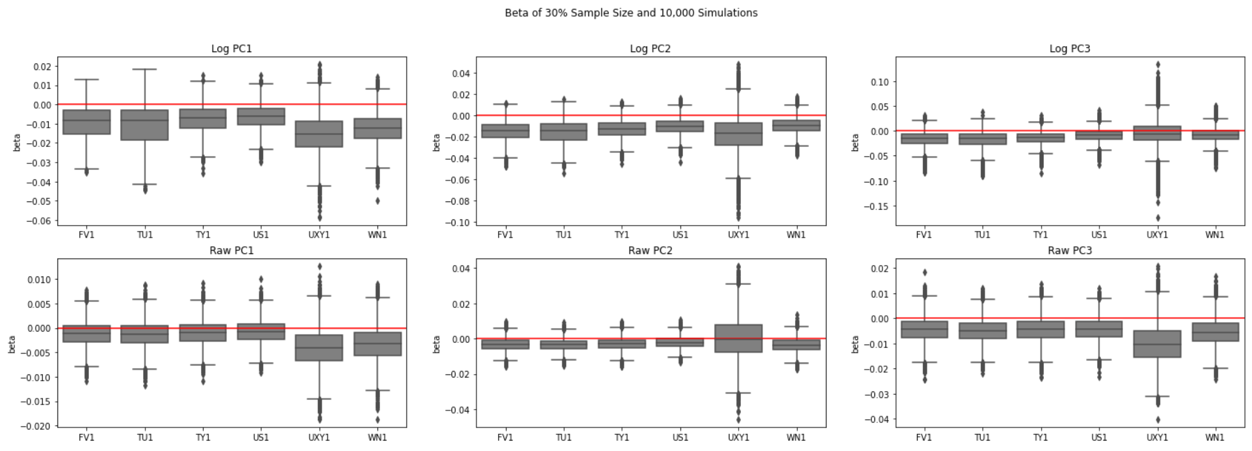
Now examine their p-values which further confirm that log PCA signal should be shorted.



This is an initial step towards confirming whether or not the signal should be shorted or be long signal. Applying the sign of the βs back onto the signal will recover the same returns found in Initial Results.

4.2 Bootstrapped OLS

To build a case that the βs aren't prone to bad sampling using a sample size of 30% and taking randomized samples and 10,000 of them a boxplot of βs can be generated.



The results show that its unlikely to get a positive β and even less unlikely to occur for log PCA.

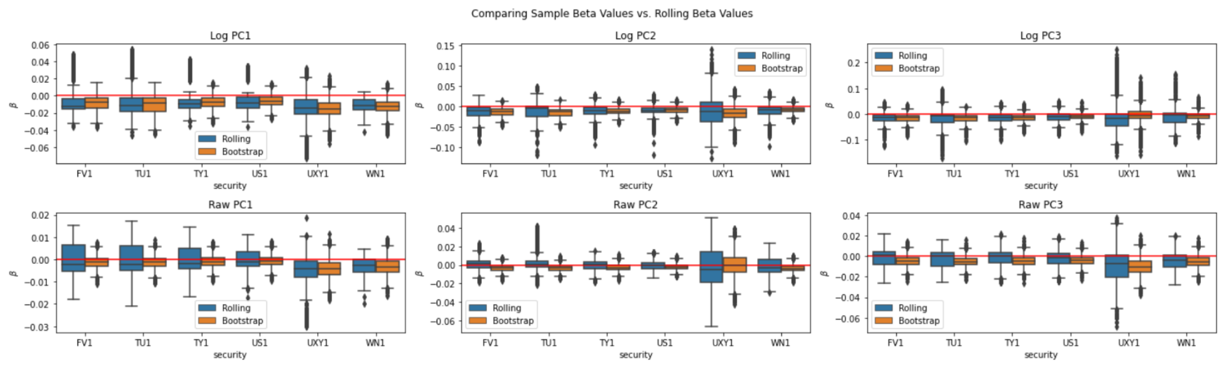
4.3 Moving Forward

At the time of writing the next step would be to run a rolling regression and then analyze betas. There are two conclusions that can be made.

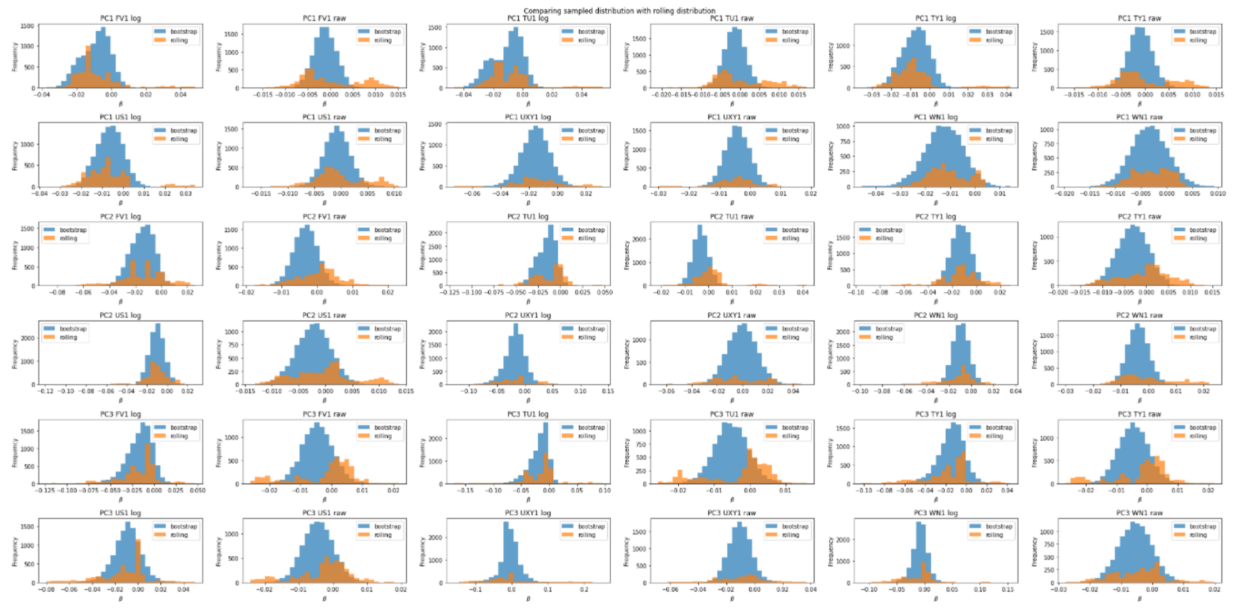
1. The rolling β relaxes the constant *long* signal or *short* signal condition
2. The rolling β is also a *specialized* bootstrap

There are two scenarios. Let's say that the rolling βs agree with bootstrap (always negative) then it just confirms the initial bootstrap findings. Let's say that the rolling βs disagree (not always negative), then it becomes a question on whether or not the rolling βs generate positive expected return. If they generate positive expected return at times then its clear that the signal isn't as *static* as originally thought of. If the signal doesn't generate positive returns then it becomes a question of the sampled data to fit the OLS.

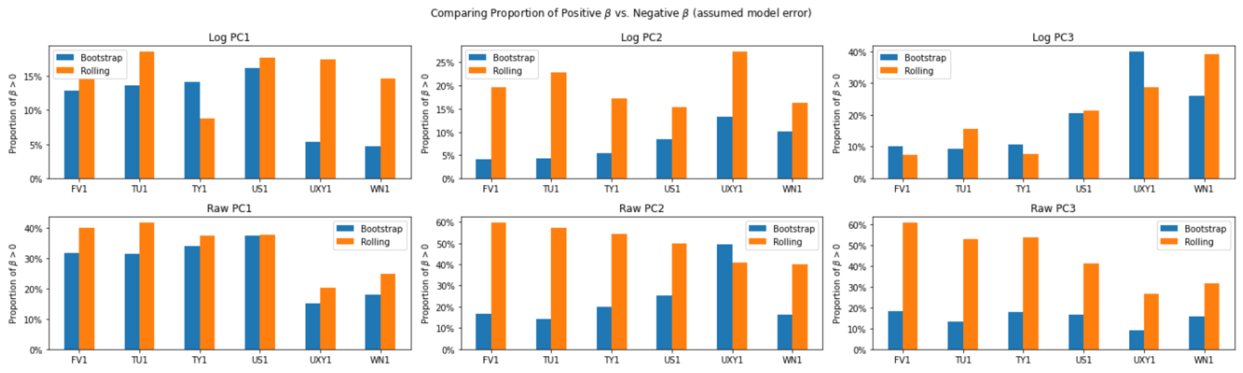
After some time it was found that a rolling OLS approach does not work well. There are two reasons to corroborate this. The first is that the distribution of rolling betas doesn't match the distribution of bootstrapped betas. The rolling betas fluctuate more from a positive to negative sign. It appears that the relationship doesn't invert since the payoff from a flip in beta doesn't generate positive PnL. Below is a boxplot comparison of the rolling betas vs the bootstrapped betas.



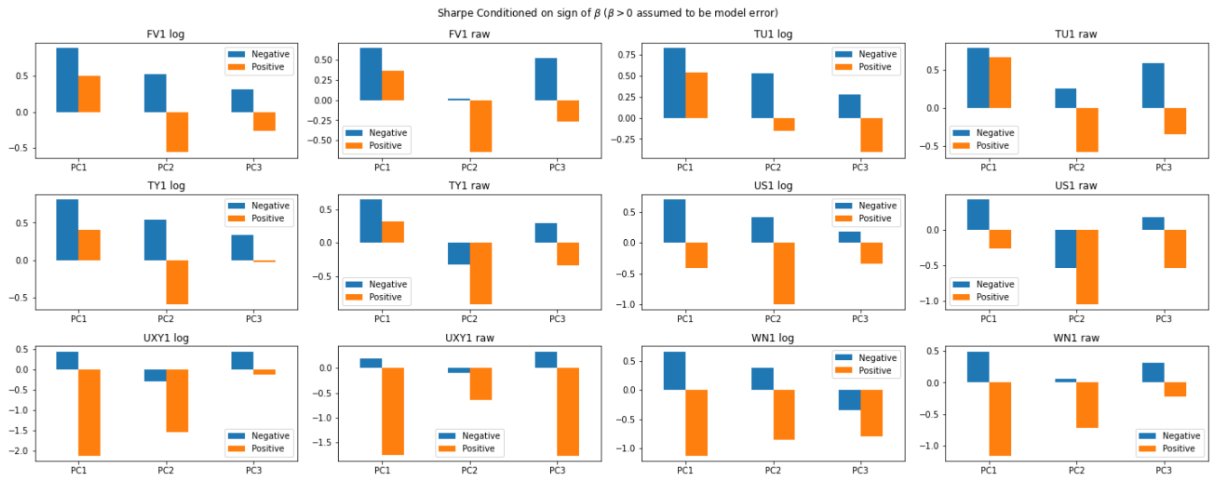
While the boxplot might imply that the rolling betas are almost as stable as the bootstrapped ones plotting the distribution shows how much they differ.



Although the distribution is important the sign of the beta truly decides how the security gets traded. Measuring the proportion of positive vs negative and comparing it to the bootstrap shows how different they are.



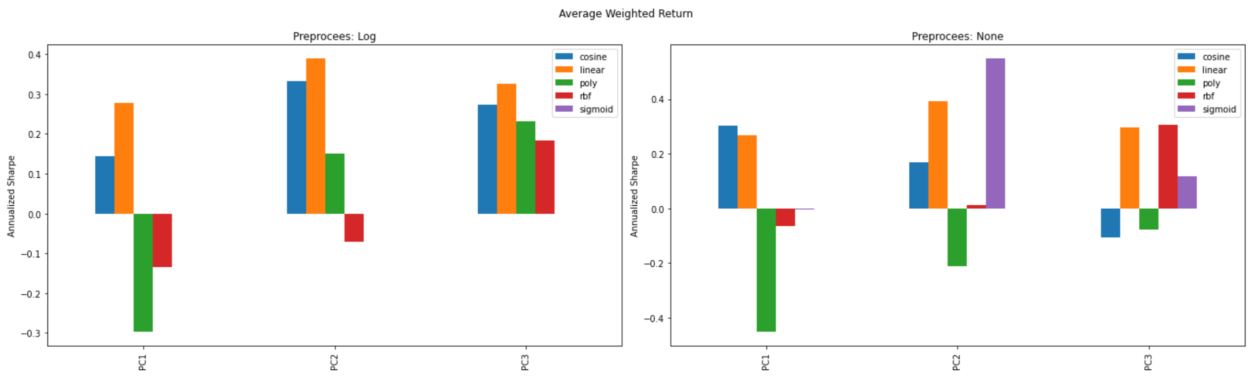
From previous tests, it is known that the signal generates positive PnL being short the signal. If the relationship actually flips (implying positive beta) then the PnL conditioned under positive beta should be positive. Below is a sharpe conditioned under the two regimes of the signal. It is evident that negative beta generates returns. There are some examples of positive PnL generated from positive betas that occur for log rates on the first principal component which may be important, but its not worth cherry-picking that model.



5 Other PCA Approaches

Throughout building this repo there have been some *alternative approaches* that have failed. Its a firm belief that if a signal doesn't show clear signs of improvement with reasonable assumptions its not worth pursuing. For example, using log rates before applying PCA generated substantial returns.

An approach that failed was to try other PCA methods. `sklearn.decomposition.PCA` has a `kernelPCA` object that has the following kernels available `cosine`, `linear`, `poly`, `rbf`, and `sigmoid`. The regular `sklearn.decompsition.PCA` default is to use `linear`. It may seem like a logical assumption to try fitting a `poly` kernel but all PCA kernels have failed. Below is a plot of the sharpes per each PCA kernel. In this case `linear` kernel beats out the rest for log rates.


Average Weighted Return
Preprocees: Log
Preprocees: None
Annualized Sharpe
cosine
linear
poly
rbf
sigmoid
PC1
PC2
PC3