Brendan O'Neil 11/11/2020

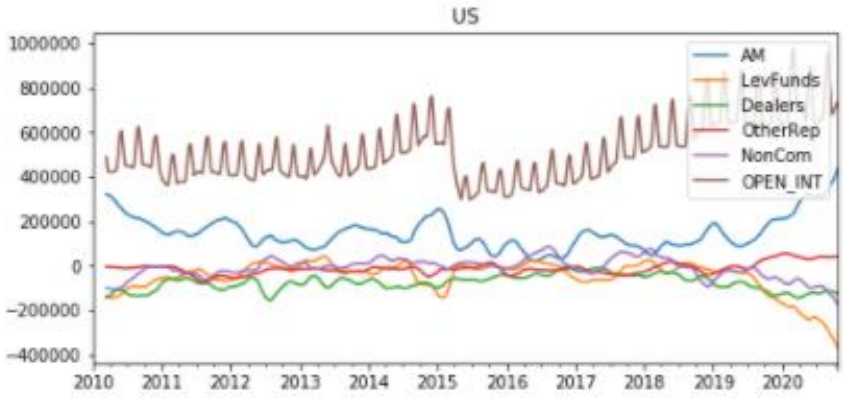
ExodusPoint Case Study – CFTC Bond Futures Data

In this document I work through my process for analyzing a dataset for predictive features, constructing signals and backtesting them.

Reading and Cleaning Data

First step is to read the data out of the excel file[1] and inspect the data for oddities using basic timeseries plots and scatterplots. The only thing that stood out to me was a sharp decline in the open interest for the US contract in early 2015, it sounded vaguely familiar and I found the CME document explaining why[2].

```
ct_dfs[ct].loc[:, ['AM', 'LevFunds', 'Dealers', 'OtherRep', 'NonCom', 'OPEN_INT']].rolling(52).mean().plot(ax=ax[0])
ax[0].legend(loc=1)
ax[0].set_title(ct)
plt.show()
```



Next I consider the swap data, given what happens to the deliverable basket for US I decide to adjust its relevant swap data to be the 15y swap rate 2010 –March 2015, and then linearly interpolate the swap maturity from 20y starting March 16, 2015 down to the 15y rate at current (using weighting determined by that interpolation, and subtract the 'roll' from all subsequent datapoints to get a clean series)[3]. I use a simple average between the 20y and 30y rate for WNs and accept 2y, 5y and 7y are close enough for TU, FV and TY, respectively.

Next, I look at a PCA and correlation matrices of the weekly changes in positioning to get a sense for the relationships involved.

Loadings Matrix:

	AM	LevFunds	Dealers	NonCom
PC1	0.591	-0.649	0.034	-0.479
PC2	-0.426	-0.048	0.809	-0.402
PC3	0.094	-0.468	0.408	0.778
PC4	-0.679	-0.598	-0.421	-0.057

[1] Code in eptest_main.py, primary function read_ep_data
[2] https://www.cmegroup.com/trading/interest-rates/mar-15-jun-15-roll-analysis.html
[3] Code in eptest_main.py, function get_interp_swaps

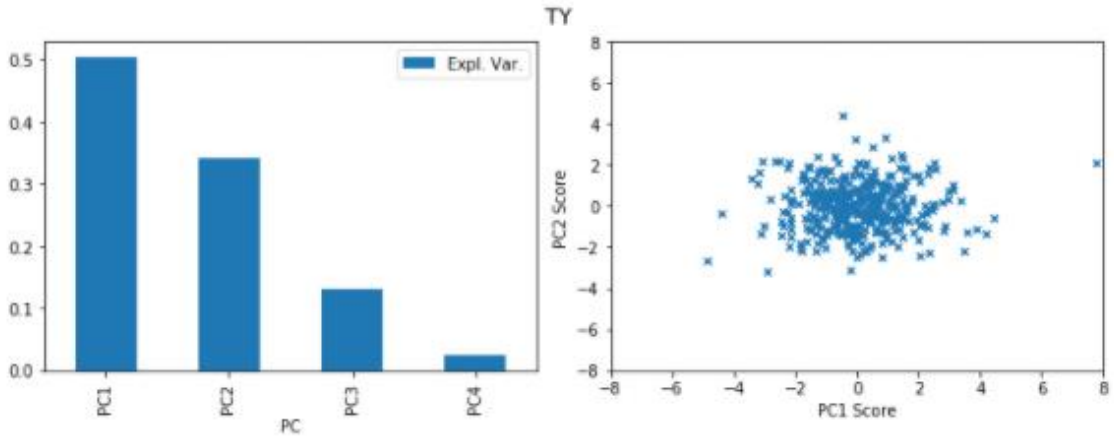
I consider Asset Managers, Leveraged Funds, Dealers and NonCommercial data since Other Reportables are relatively small and Commercial is close to just the other side of NonCommercial positioning. NonCommercial is considered as a proxy for speculators in general with respect to the CoT data.

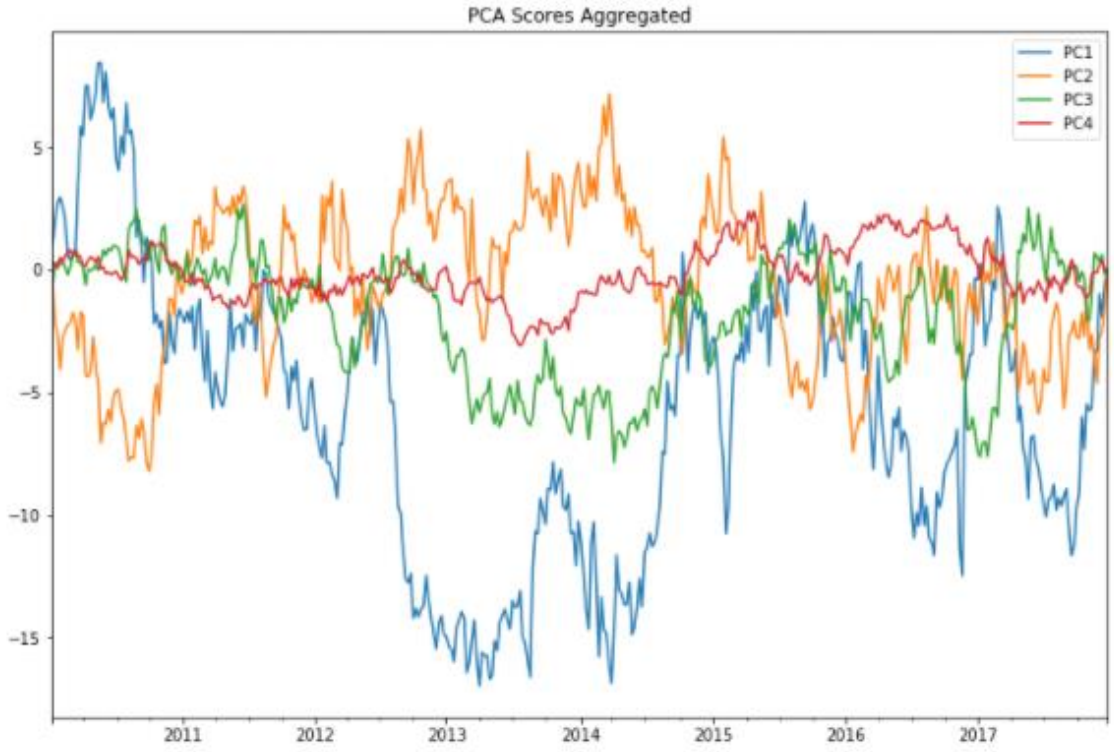
With the TY contract, for example, PC1 and PC2 explain around 85% of the variance in positioning changes. One might interpret them as:

PC1: Asset Manager changes in positioning are offloaded to speculators

PC2: Buyside manager changes in positioning are taken down by dealers

We will find later issues with collinearity in the data, even after removing some of the market participants, and techniques like principal components regression are commonly used to deal with this issue. In the interest of time and maintaining a better intuition in the data I decided to go another route.

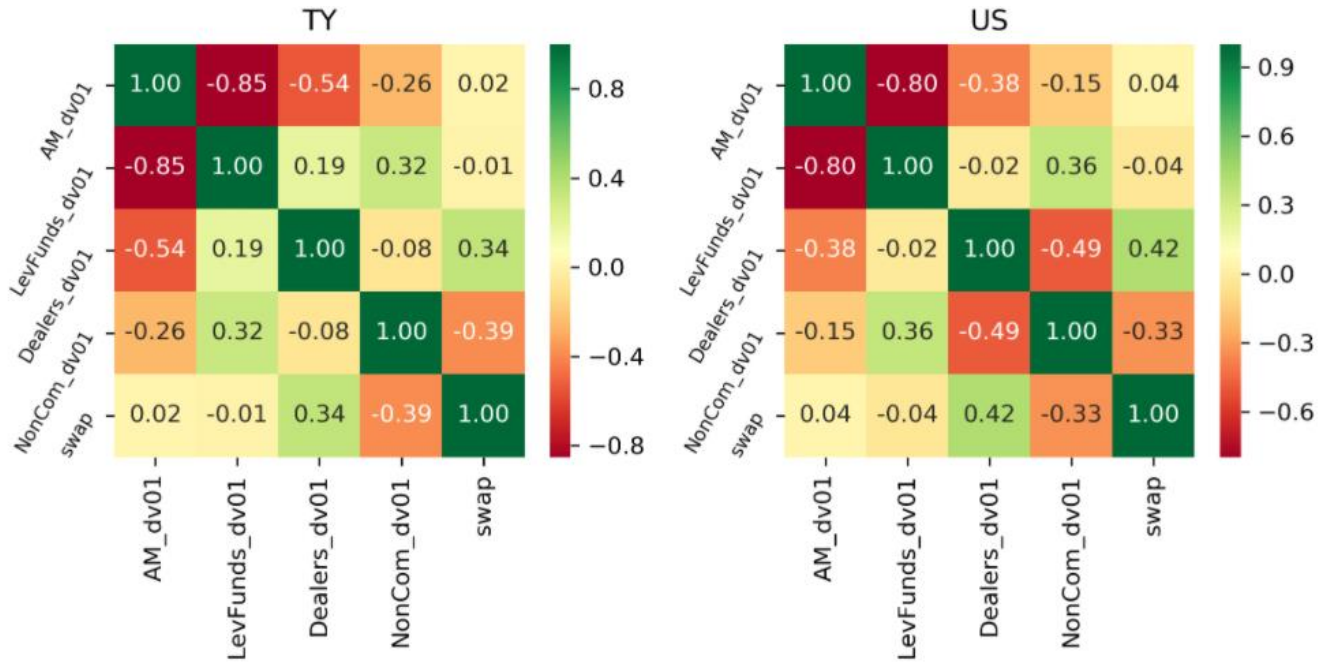




Initial Contemporaneous Analysis

The first question I would like to answer is whether the CoT positioning data has significant explanatory value in analyzing contemporaneous changes in yields (as proxied by swaps). Before I start looking at the data more closely I remove all of the samples from 2018 onward- I will want an untouched out of sample period to test my in-sample findings on unseen data, about 25% of the data is reasonable.[4]

For starters I want to normalize my data to some extent to make it more comparable across contracts. I try both weighting the positions by dv01[5] and as a percentage of the average trailing open interest[6]. For the rest of the work I consider the dv01 weighted values[7]. For all contracts looking at weekly changes in positioning, we get the relationships we might expect- changes in AM positioning are taken down by speculators and dealers on average.



Ordinary Least Squares

I regress 1wk Tuesday changes in the relevant swap series for each contract against its corresponding futures contract positioning changes from AM, LevFunds, Dealers and the swap change lagged 1 week, to account for autocorrelation of changes in yield. As we see below, there might be some relationship with changes in dealer positioning and the lagged change, but there is clearly large collinearity issues, so we can only conclude that the t-statistics are not robust and the betas are very unlikely to be

[4] Preferably I would also want samples from different 'regimes', e.g. in the beginning and middle of the dataset, but in the interest of time and since the data is weekly and only ~11 years post-GFC, I just reserve the end. I use another technique called cross-validation to 'stretch' the in-sample period to consider quasi-out of sample periods.

[5] Contracts*Dur*px/100*ct_size/10e4

[6] To mollify the periods of open interest on the front contract that are after the roll is mostly complete

[7] Obviously does not matter within contract at the same time but hopefully introduces some consistency over time. Also, I wanted to explore cross-contract dv01 relationships, but ended up settling on z-score changes anyway

numerically stable. As discussed before, we could try principal components regression to deal with this, but I prefer to use a regularization technique like Ridge or Lasso.

OLS Regression Results

Dep. Variable:	7y	**R-squared:**	0.124
Model:	OLS	**Adj. R-squared:**	0.113
Method:	Least Squares	**F-statistic:**	11.55
Date:	Wed, 11 Nov 2020	**Prob (F-statistic):**	1.89e-10
Time:	12:03:53	**Log-Likelihood:**	382.44
No. Observations:	414	**AIC:**	-752.9
Df Residuals:	408	**BIC:**	-728.7
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0025	0.005	-0.519	0.604	-0.012	0.007
AM_dv01	-4.361e-10	1.64e-09	-0.267	0.790	-3.65e-09	2.78e-09
LevFunds_dv01	3.407e-09	1.62e-09	2.099	0.036	2.16e-10	6.6e-09
Dealers_dv01	7.967e-09	2.02e-09	3.940	0.000	3.99e-09	1.19e-08
NonCom_dv01	1.846e-10	1.05e-09	0.176	0.860	-1.87e-09	2.24e-09
7y_lag1	-0.0357	0.047	-0.761	0.447	-0.128	0.057

Omnibus:	29.354	**Durbin-Watson:**	1.990
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	56.560
Skew:	0.422	**Prob(JB):**	5.23e-13
Kurtosis:	4.602	**Cond. No.**	8.48e+07

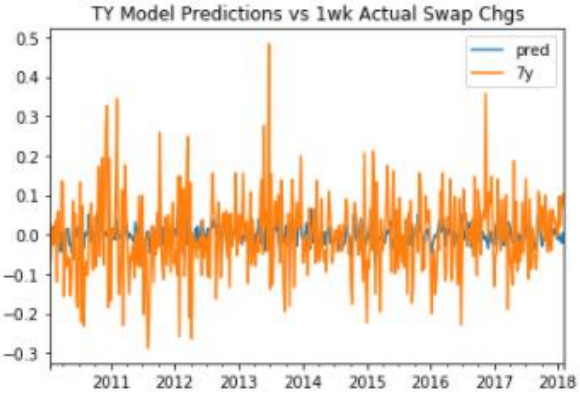
Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 8.48e+07. This might indicate that there are
strong multicollinearity or other numerical problems.

Lasso

I like to use Lasso in these cases. It is different from OLS in that it penalizes each beta in the objective function so that the betas are shrunk until their explanatory power outweighs the penalization. This helps significantly in cases such as this, where for example AM and LevFunds positioning are highly negatively correlated. Lasso also behaves like a simple feature selector in that it will shrink betas to 0 for data it does not find useful out of sample. How much penalization to use is evaluating using a time series cross validation process, where the model is fit on a rolling window and evaluated on a subsequent period of (somewhat) out of sample data. The penalization parameter that best fits the out of sample

periods on average is selected. The penalization (regularization) parameter is called alpha, in the charts to the right increasing alpha means more penalization of betas. For this to work properly we first need to normalize our features to z-scores so that the penalization affects them equally. [8]

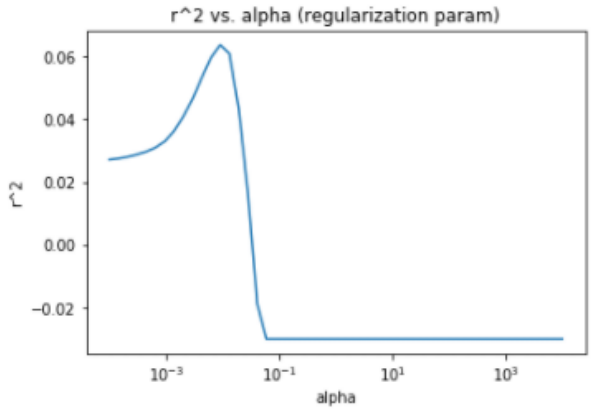
In the case of 1wk changes in TY positions and swap rates on the right, the betas load negatively on AM (increase in AM positioning implies rally) for -1.5bps per z-score AM positioning change, and positively on Dealers for 1.2bps. Note how small the predictions are (in blue) versus the actual changes in orange below— r^2 is very low.



For 5wk changes Dealers and Specs are selected but the results are very poor. Nothing does better out of sample than just taking the simple average of swap changes over the training period. This is incidated in a negative r^2 and 0 betas for the best model.

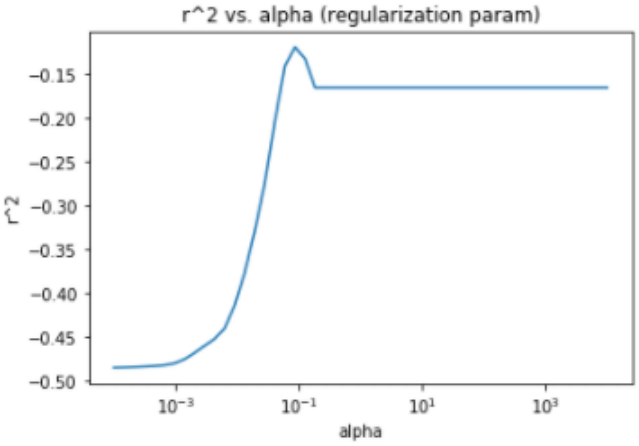
Other contracts exhibit similarly poor concurrent results.

TY 1wk changes




```
best_score           0.0636896
alpha               0.00910298
best_score_ratio      -2.12482
Name: TY, dtype: object

AM_dv01          -0.015190
LevFunds_dv01     0.000000
Dealers_dv01      0.012419
NonCom_dv01       0.000000
7y_lag1          -0.000000
Name: beta, dtype: float64
```

TY 5wk changes



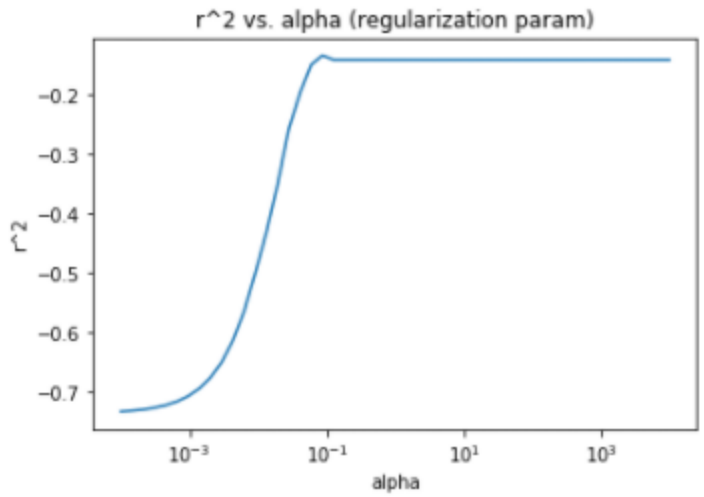

```
best_score           -0.11849
alpha               0.0868511
best_score_ratio     0.718053
Name: TY, dtype: object

AM_dv01          -0.0
LevFunds_dv01     0.0
Dealers_dv01      0.0
NonCom_dv01      -0.0
7y_lag5           0.0
Name: beta, dtype: float64
```

[8] This code is housed in classification.py, main function run_cv

Forward-looking Regressions

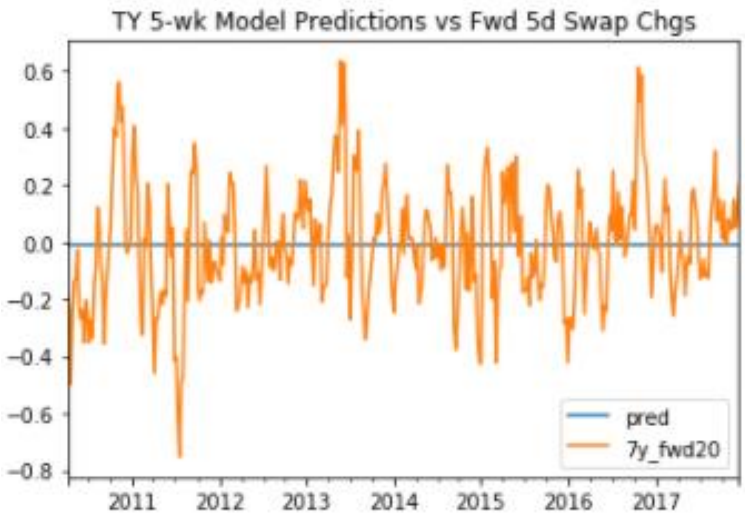
The next question is whether changes in positioning can predict future changes in swap rates. Since the coincident results were quite weak, I believe if there is going to be predictive power it will result from accumulations of positions over a rolling window. First, CoT data is collected on Tuesdays, but generally reported on Fridays. For this to be of value in predicting future changes I need to align the data with when it is reported, for simplicity I will shift it 3 business days forward in time. While the 1wk changes in positioning are clearly stationary, as we take changes over longer timeframes we are likely to run into issues with stationarity- in short we want to avoid the problems associated with non-stationary data and spurious regression. To analyze the stationarity of various window lengths of positioning data and swap rate changes, we apply augmented Dickey-Fuller tests[9] to evaluate whether the data is statistically different from a random walk. For brevity I do not show these here, I find that between 13 and 26 weeks the ADF p-values start to trend into the single percentage points, so I set a cutoff at not analyzing more than 13wk changes to be safe.

TY 5wk changes vs 20d fwd swap changes



For all contracts, almost no combination of CoT data, change sampling frequency nor subsequent swap rate change window performs meaningfully different from guessing the average. While I could mine the data for some relationship, the fact that it is not even somewhat consistent suggests to me there is little there.

```
best_score          -0.135106
alpha               0.0868511
best_score_ratio     0.948531
Name: TY, dtype: object

AM_dv01         -0.0
LevFunds_dv01    0.0
7y               0.0
7y_lag5          0.0
Name: beta, dtype: float64
```



[9] Code in eptest_main.py, function test_adfuller calls statsmodels function

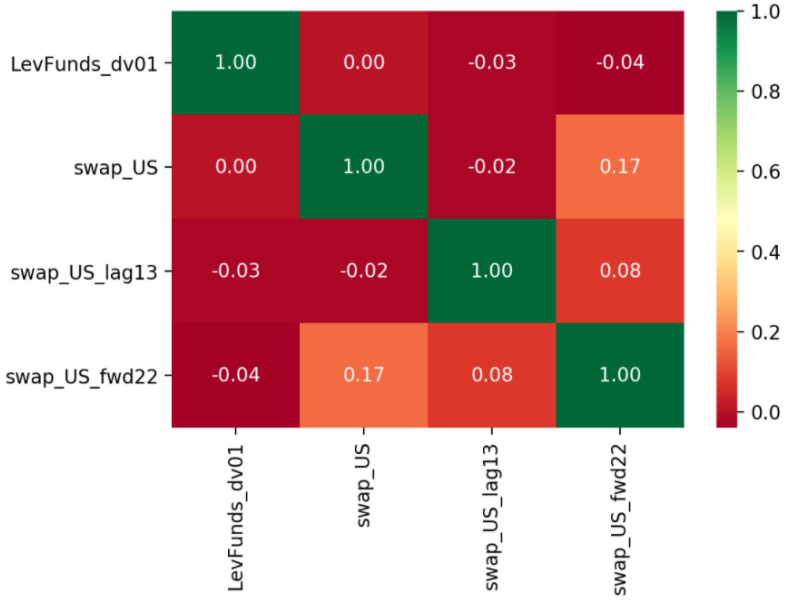
Extreme Value Analysis and Classification

Small changes in positioning might be contributing too much noise to the analysis. We can try filtering for only large changes in positioning data. We are also not necessarily interested in predicting the value of future changes in swap rates, but most importantly whether they rally or sell-off. For this purpose, I change the problem to be analyzed to predicting the sign of the future change, *weighted by the absolute value of the change*[10]. The intuition here is that we are interested in whether we can profit from our predictions, and we want to assign more weight to larger profits (and penalize larger losses). I use a simple logistic regression.

I run the same type of cross validation process with the filtered data for the adjusted problem across all the contracts on a few sets of position change windows and future swap change periods. The results are not particularly exciting, if there is any value in this data for outright duration it is not clear how different it is from a coin flip. Below is a snap of the results of trying to predict the sign of subsequent 1mo swap changes using the normalized 13wk changes in LevFunds positioning, filtered for when it is greater in magnitude than +/- 1.0 z-score. Changing the forward window, the length of positioning changes data the filtering z-score and which positioning data to use yield wildly different betas and weighted accuracy scores[11]. I will move forward, with reservations, using this setup for the US contract because the betas are intuitive- the betas load positively on past changes in rate and negatively on past changes in LevFunds positioning (so an increase in LevFunds correlates to future rallies in yield). This suggests there might be a momentum indicator value, though weak considering FV and TY are worse than coin flip and WN is barely any better.

	FV	TU	TY	US	WN
beta	[[-0.01121, -7e-05]]	[[0.31264, -0.03243]]	[[-0.04376, 0.03671]]	[[-0.13728, 0.51074]]	[[-0.12599, 0.01361]]
best_score	0.48824	0.524864	0.489247	0.603659	0.525836
C	0.00910298	0.828643	0.0596362	3.72759	0.126486
best_score_ratio	0.960472	0.885313	0.978494	1.50915	1.26403

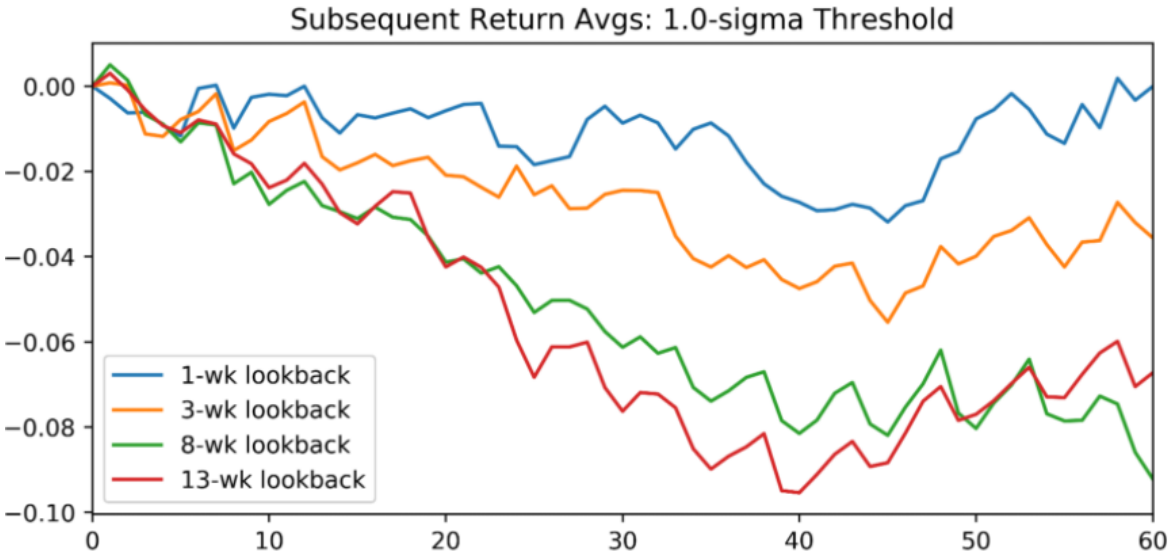
To the right is the correlation matrix for 13wk changes in LevFunds z-score, 13wk changes in the interpolated swap rate corresponding to the US contract, the same value lagged 13wk and the forward 22d swap change, filtered for when the LevFunds change z-score is above +1 or below -1.

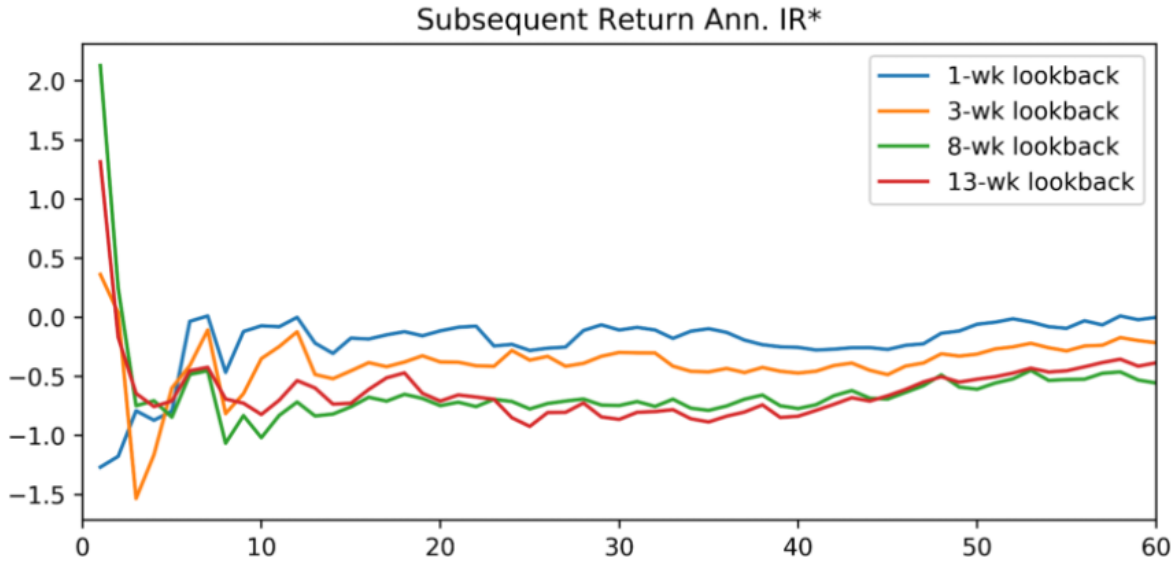


[10] Code in classification.py, main function run_cv_weighted, implements my own cross-validation

[11] Accuracy being whether the algorithm predicted the correct future sign of swap rate changes

Below is the same setup, showing the average changes in yield after the threshold is hit, up to 60 trading days. It confirms, in-sample, that an 8-13wk change in US LevFunds positioning greater than +/- 1.0 z-score on average is followed by a rally/sell-off in the 15-20y swap rate over the next couple months.





lookback	1			3			8			13		
fwd period	5	20	60	5	20	60	5	20	60	5	20	60
Mean	-0.0116	-0.0058	-0.0000	-0.0078	-0.0209	-0.0356	-0.0131	-0.0413	-0.0922	-0.0109	-0.0424	-0.0672
Std Dev	0.1038	0.1779	0.3329	0.0924	0.1956	0.3372	0.1093	0.1952	0.3383	0.1089	0.2115	0.3562
Ann. IR	-0.7933	-0.1145	-0.0002	-0.5988	-0.3779	-0.2155	-0.8452	-0.7477	-0.5562	-0.7082	-0.7092	-0.3853
Min	-0.2890	-0.5847	-0.9923	-0.1938	-0.5847	-0.9923	-0.3950	-0.5847	-0.9923	-0.3950	-0.5847	-0.9923
Max	0.3120	0.3795	0.8069	0.3120	0.4545	0.6615	0.2468	0.4655	0.7620	0.2403	0.4655	0.8069
Max DD	-0.2890	-0.5847	-0.9923	-0.2098	-0.5847	-1.0571	-0.3950	-0.6340	-1.0072	-0.3950	-0.6340	-1.0072
Num Obs	108.0000	108.0000	108.0000	91.0000	91.0000	91.0000	108.0000	108.0000	108.0000	111.0000	111.0000	111.0000
Hit Ratio	0.4444	0.4722	0.5370	0.4615	0.4945	0.5055	0.4444	0.4630	0.3889	0.4775	0.3874	0.4144

The betas (right) of the logistic regression are inconsistent over time, which is concerning. This implies we would not have seen the same relationship at different points in time, so that if we were to have tried to run this strategy, we would be making very different decisions in real time.

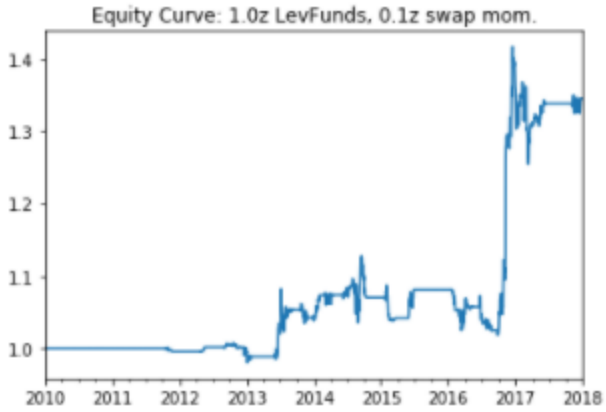
	LevFunds_dv01	swap_US
2012-03-16	0.573146	-0.177329
2013-05-24	0.280014	-0.034414
2014-08-22	-0.096083	0.329840
2016-03-11	0.062586	0.123189
2017-01-13	-0.060063	0.341244

Backtest

To backtest the results of this research, I build an indicator series composed of:

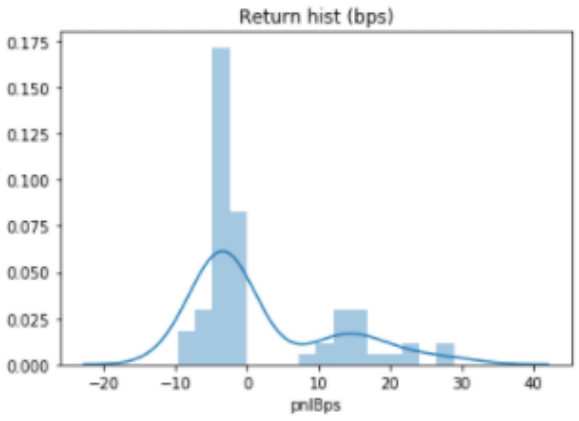
- Sign of the trailing 13wk LevFunds z-score filtered > +/- 1.0z AND
- Sign of the trailing 13wk change in swap rate filtered > +/- 0.1z

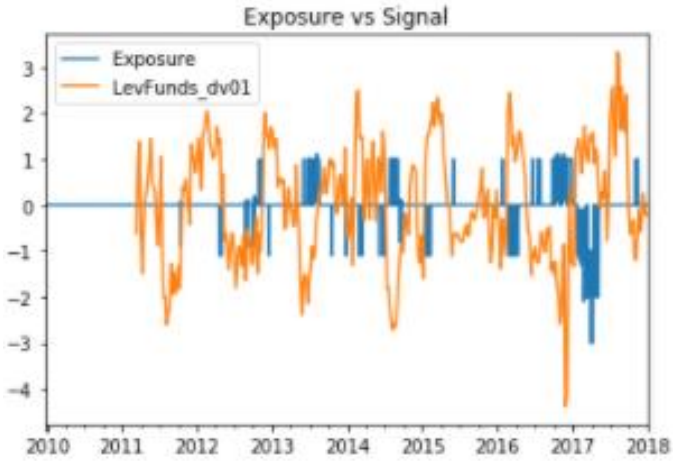
To trade this proxy for momentum, we will go 'long' the swap rate (short duration) when the LevFunds z-score is below -1.0 and the sign of the trailing 13wk swap rate change is positive, and short when the inverse is true. On the first pass, we hold the position for 40 trading days and close. This equity curve is to the right.



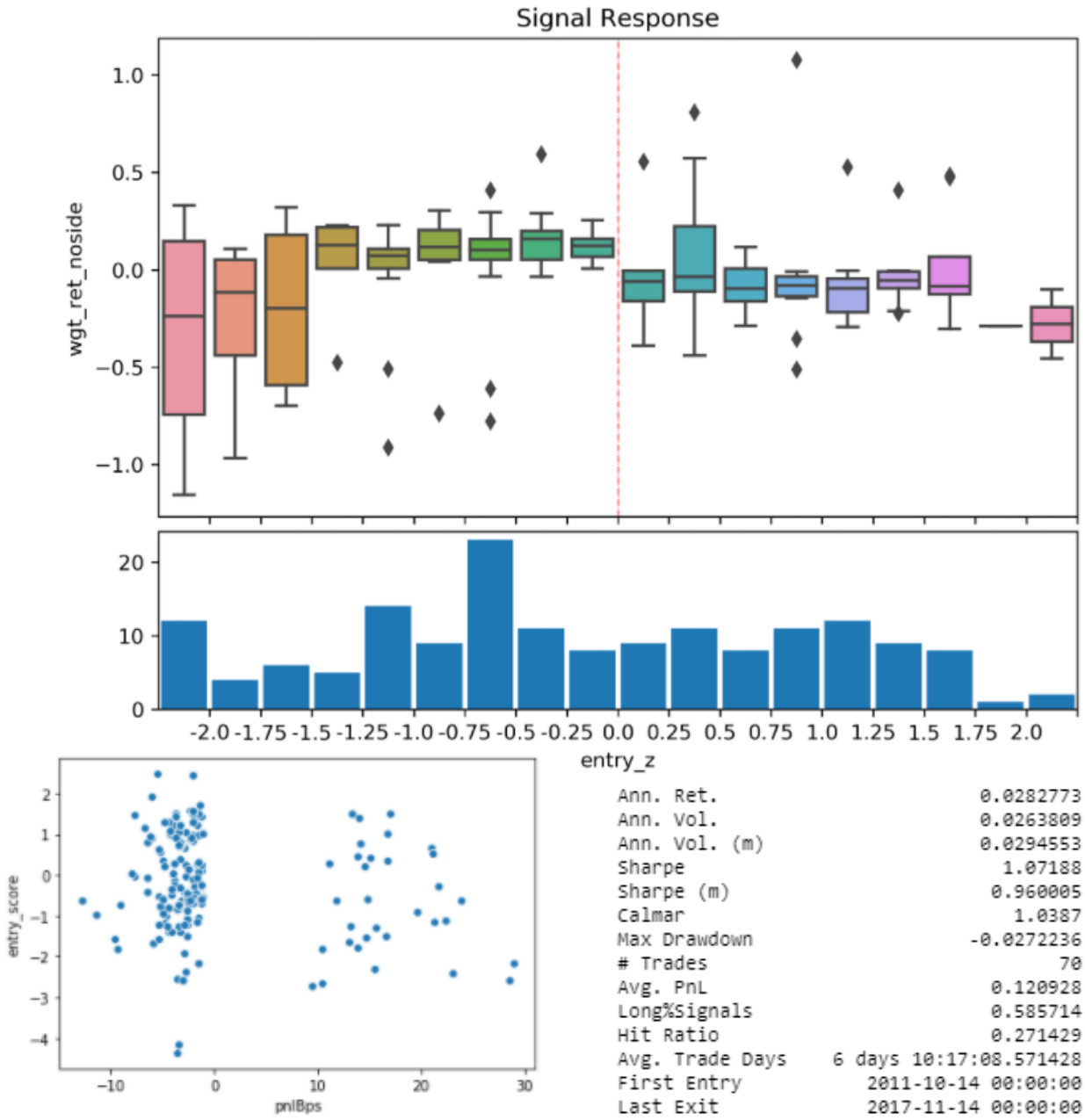
Since it is a momentum type strategy, I set some take profit and stop loss rules on the strategy- when a trade is up 3x the daily std of 6m trailing rate moves, I take profit, and when it is down more than 0.25x the trailing vol, I cut losses (at the close rate, so the loss will be worse). It performs marginally better in sample.

Clearly the strategy has a high positive skew with most trades losing a little and a handful of big winners.





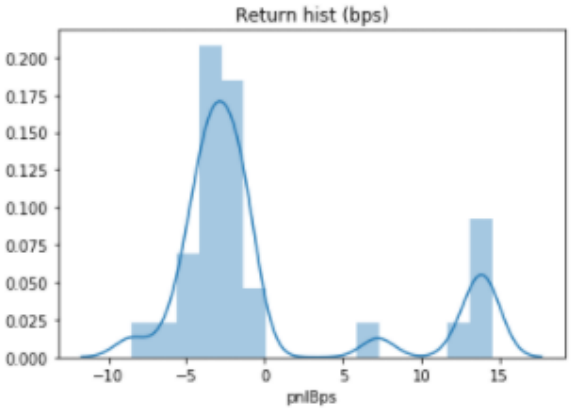
Looking at the summary statistics (bottom right), we see mostly positive results with a low hit rate, as expected since we optimized the strategy in-sample. We also see some interesting results from the scatter and signal-response plot[12] of PnL against the entry signal z_score of LevFunds positioning changes (I relax the signal threshold to produce this). In an ideal scenario, we would see an S-curve from the bottom left to top right, indicating that highly negative entries correspond to larger negative future returns (that we would profit from by being short), and conversely that highly positive signals correspond to larger positive future returns. In this case we see that a bulk of the PnL is generated from large signal shorts, and large positive signals produce the worst bucket of strategy returns on average.



Ann. Ret.	0.0282773
Ann. Vol.	0.0263809
Ann. Vol. (m)	0.0294553
Sharpe	1.07188
Sharpe (m)	0.960005
Calmar	1.0387
Max Drawdown	-0.0272236
# Trades	70
Avg. PnL	0.120928
Long%Signals	0.585714
Hit Ratio	0.271429
Avg. Trade Days	6 days 10:17:08.571428
First Entry	2011-10-14 00:00:00
Last Exit	2017-11-14 00:00:00

[12] The signal response plot is a boxplot showing the mean and distribution of strategy outcomes, bucketed by the signal z-score, weighted by how unique the trade is (e.g. if two trades triggered back-to-back they would be weighted ½ each). I flip the sign of the negative signals so that the ideal outcome is an S-curve rather than parabola

Out of Sample Analysis and Conclusion



Ann. Ret.	-0.00477703
Ann. Vol.	0.0181545
Ann. Vol. (m)	0.0069227
Sharpe	-0.263133
Sharpe (m)	-0.690053
Calmar	-0.116208
Max Drawdown	-0.0411076
# Trades	30
Avg. PnL	-0.0481219
Long%Signals	0.733333
Hit Ratio	0.2
Avg. Trade Days	7 days 10:24:00
First Entry	2018-01-26 00:00:00
Last Exit	2020-10-22 00:00:00

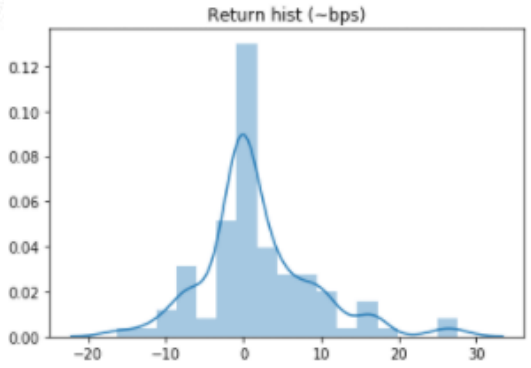
The out of sample results are quite poor. There is a similar return profile but not enough winners to make up for the many small losers. In the end this is not particularly surprising given the in-sample results were not very stable under small modifications to the parameters. This is an important point with respect to avoiding overfitting a backtest- if there is the relationship between the data and future returns that is hypothesized, the strategy should do similarly well in the local area of the parameter space. This was not the case. Prior to setting take-profit and stop-loss conditions on the strategy, almost all the PnL was comprised of the days after the 2016 US elections.

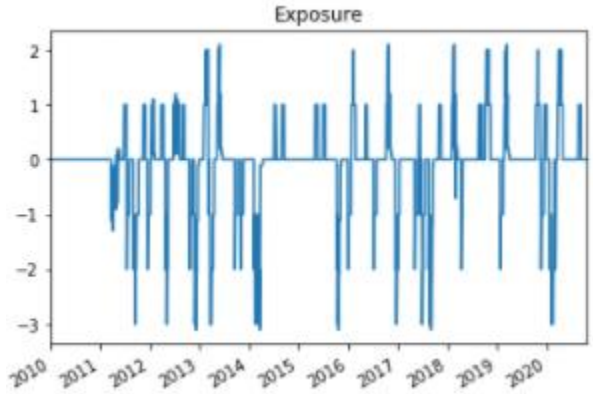
I found the same issues with coming up with a relationship to backtest in the first place- the regression betas were not stable over time for the same contract, or between contracts over the full data set. Simply looking at scatterplots of various positioning indicators versus forward returns suggested there would be little substance. There was not even a particularly meaningful relationship between contemporaneous changes in positioning and yields (although previous changes in swap rates seemed that they might have predictive power over changes in AM positioning). Though careful to use rigorous methodology for evaluating relationships in a cross-validated framework, I was particularly guilty of multiple testing bias to identify any positioning indicator that might produce meaningful in-sample results. One can be rigorous in the practice of evaluating relationships on a rolling out-of-sample basis, but if you run 20 tests on the same data you have effectively completely polluted that check on your research. Further, I think one needs to consider that if the hypothesis is correct, why the opportunity has not been arbitraged away by market participants.

Finally, I tried similar methodologies for evaluating positioning signals for trading swap curves which were equally uninteresting. The next idea I would have looked at is using positioning to indicate when to close out a trade, e.g. if the signal flips sign, or trading butterflies. After that, I would have tried to use the positioning data to improve a more basic momentum strategy. The idea there would be to develop a momentum strategy and then use classification algorithms to evaluate whether a model based on positioning data can predict whether to lever up, scale down or cut a momentum trade early. This is a slightly different problem than predicting changes in swap rates though the same methodology is used, the target variable is then the returns of a momentum strategy. I think, however, that if there was value in the data for trading duration or curves it would have been relatively clear by now.

Appendix

I figure I would include the results from a similarly overfit strategy on the US-FV curve, taking z-scores of the US and FV changes in LevFunds positioning changes over trailing 5wks and trading a momentum strategy off of it with no take-profit/stop-loss. Included in the sandbox-signals notebook under the "curve" headings.





Ann. Ret.	0.0192832
Ann. Vol.	0.0260829
Ann. Vol. (m)	0.0281714
Sharpe	0.739303
Sharpe (m)	0.684495
Calmar	0.52342
Max Drawdown	-0.0368407
# Trades	98
Avg. PnL	0.0906
Long%Signals	0.540816
Hit Ratio	0.530612
Avg. Trade Days	18 days 15:11:01.224489
First Entry	2011-03-25 00:00:00
Last Exit	2020-09-10 00:00:00

Notes on code:

- Most of the presentation material can be found in the python jupyter notebook under ExPtTest named sandbox-signals, including some other avenues I pursued.
- A substantial portion of the backtest evaluation code I had already written for my own understanding while independently studying Marcos Lopez de Prado's book *Advances in Financial Machine Learning* over the past year or so. It is housed primarily in Research.signals.py, Preprocessing.etf_trick, Preprocessing.labeling, and Preprocessing.sampling. Since the backtest here was straightforward I could have written simpler code to achieve the same thing, but I figured I had it so I would use the time on other pieces of the presentation. For this reason, there's a substantial amount of largely irrelevant code included in labeling.py, sampling.py and signals.py.
- The regression and classification 'machine-learning' type code is housed primarily in Modelling.classification.py, some of this I had written before, but a substantial portion for fitting and scoring with sample weights I wrote to use on this project- I had been meaning to for some time because sci-kit learn does not implement that and it's pretty useful for markets research.

Decision Boundaries

Below are some decision boundaries from logistic regression and support vector classifiers on predicting the (weighted) sign of future swap moves. In blue are subsequent 1mo sell-offs, in red are rallies, X markers are test set (2016-2017), circles are training set, size of markers represents the absolute size of the move. The gradient represents probabilities and the SVC decision around -1 or 0, respectively