Data Mining Techniques Applied to Econometric Panel Data for FX Forecasting

Charles Naylor

Contents

Introduction

Summary of Currency Markets

Macroeconomic forecasting is generally considered to be one of the hardest challenges in finance. In comparison with forecasts on stocks or bonds, currency (a.k.a "foreign exchange", or FX) price movements react more purely to the flows of global trade and geopolitical risk, because they have no intrinsic value. A currency's worth is quoted only relative to other currencies.

The Carry Trade

That said, there is still a basis for investing in currencies in that any holding will earn interest, just like a savings account at a bank. In the USA, the 3-month interest rate is about 2.3%. In Turkey, it's about 17%. Other things being equal, an investor could borrow money in American Dollars, then lend it out in Turkish Lira, and earn the difference between these rates for a nearly 15% annual return. This is called the **carry trade**.

The carry trade is not riskless because the spot rate, i.e. the number of USD that a Turkish Lira can buy, is not fixed over this period. In fact, undergraduate economics classes still teach that any profit possible from this trade will be neatly ironed out due to spot price movements, a phenomenon known as **Covered Interest Rate Parity**. The persistance of the carry trade in the face of theory is a reminder that economic models elide substantial frictions experienced in the real world.

Drivers of Returns

In FX, that interest rate differential, the "carry", is relatively stable but subject to punctuated equilibrium as new data comes to light. In developed markets, this data primarily consists of central bank rate decisions and the economic news that might affect those decisions. Price movements, however, are the deterministic result of countless iterative, interacting agents. While we may know many of these agents' motives, it is not possible to aggregate their behavior with any accuracy, because the actions of each agent are affected by those of all of the other agents, and small measurement errors compound. For example, it is impossible to tell the periodicity of market data without context. A day's worth of price movements at 5-minute increments looks the same as a year's worth of daily movements. The asset price measures the result of a chaotic, nonlinear dynamical system: a scale-free network (Barabasi 2002).

Motivation

Speculators in currencies try to pocket the carry while avoiding the risk of major movements in currency prices. The strategy has been likened to picking up nickels in front of a steamroller. Thus, in spite of the difficulties inherent in forecasting currency movements, a *successful* speculator must have some idea of what those movements will look like in the future.

There are many possible methods available to create these forecasts. Professionally, econometric panel data (i.e. the same set of economic measurements repeated for multiple countries) has been plugged into a Kalman Filter in order to capture the evolving relationship between indicators and their currencies, while also recognizing that one currency's movements will affect all other currencies. The Kalman filter has the advantage of having a closed-form solution and being well-adapted to testing in systems in which one expects to add new data regularly. In its most basic form, the disadvantage is that a Kalman filter requires careful tuning of the relationship between its output variables, and of the covariance of evolution of its factor weightings.

Prior work has been done on the same data set as a case study of generative Bayesian forecasting techniques (Naylor 2017). The study applied a Gaussian Process Regression to panel data for 11 of the most traded world currencies. The result was a well-specified and validated forecasting model whose error term was far larger than its signal. This jibed with the results seen professionally using a Kalman Filter, but thanks to posterior predictive checking the flaws are glaringly obvious. Can this, or a similar forecast, be improved?

Work is being done at the University of Maryland into the application of neural network reservoirs to chaotic systems (Pathak et al. 2018). After surveying some simpler machine learning techniques, this project will apply these techniques to the residuals of the Gaussian Process regression, or in combination with some other regression on the panel data.

Goals

- Run the panel data through a set of standard data mining techniques to provide a baseline.
- Apply reservoir computing techniques to existing or new regressions.

The Data

```
packages <- c("tidyverse", "scales", "lubridate", "ggthemes", "randomForest", "cluster", "e1071", "abin
lapply(packages, function(X){suppressPackageStartupMessages(library(X, character.only=T))})
```

Data consists of the weekly currency returns, plus a raft of weekly economic data.

Initial Data Gathering

While efforts were made to use free data as much as possible, even after extensive clean-up, much of it was of inadequate quality. Even proprietary data from Bloomberg's professional API took some work to put into proper format. Shown below is a digest of some of the tribulations undergone in putting the economic data in order. The full experience can be examined in the documentation for (Naylor 2017).

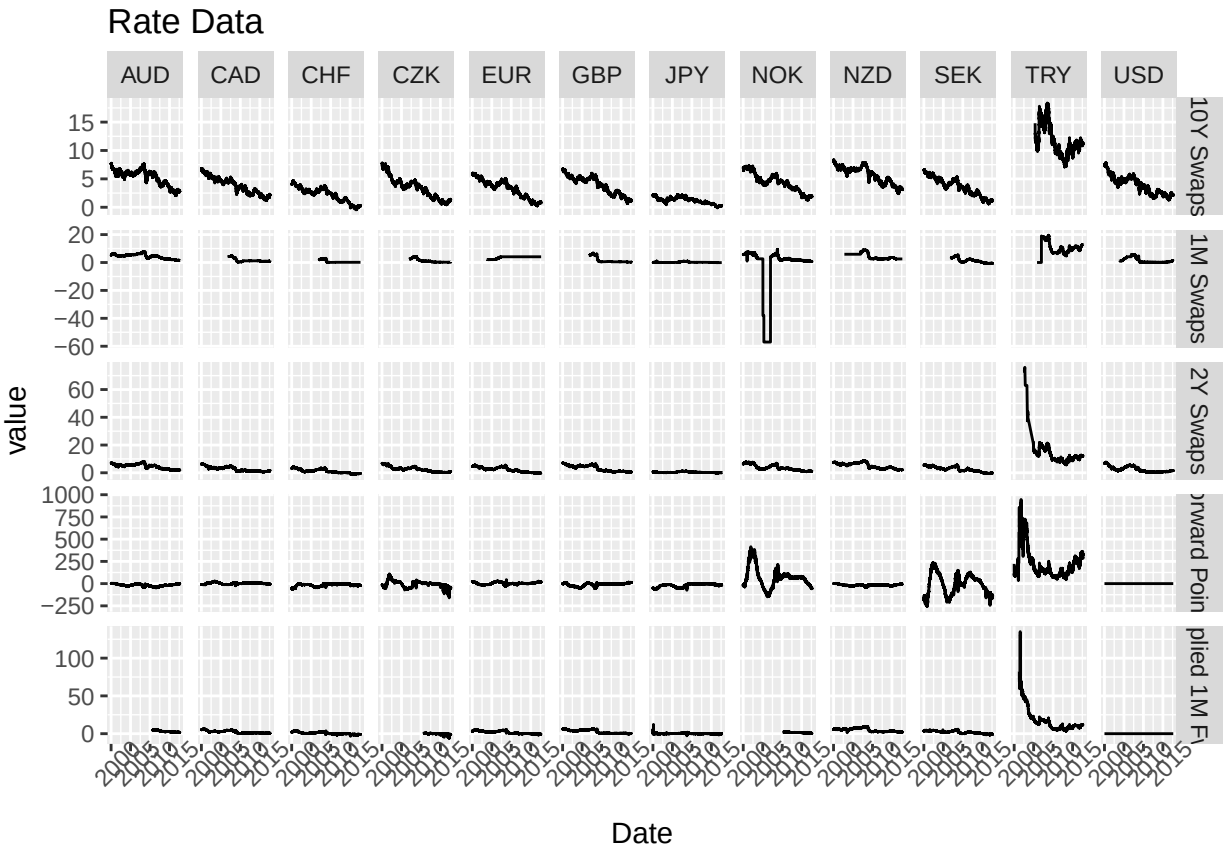
Swap Rates

As mentioned before, returns on FX consist of two elements: changes in the spot rate, i.e. the exchange rate for physical currency, and the interest rate differential between the two currencies in the pair. In this study, all returns were considered versus the USD.

Swap rates are the interest rates offered on short-term debt instruments known as Swaps, and are often used as a proxy for the interest rate differential. The ideal source would be the implied 1M forward rate, which is literally what investors will receive. There were a few options to check.

Read in raw data from Bloomberg.

```
library(readxl)
rate_pages <- c("10Y Swaps", "2Y Swaps", "Forward Points", "1M Swaps", "Implied 1M Fwd")
RATE_PATH <- "data/proprietary/rates.xlsx"

rates <- map_dfr(rate_pages, read_xlsx, path=RATE_PATH, na="#N/A",
                 col_types=c("date",rep("numeric",12)), .id="type")  %>% #read all worksheets on the xl
         mutate(type=factor(rate_pages[as.numeric(type)])) %>% #replace numeric id with the rate page n
         gather(asset, value, -type, -Date, factor_key=T) %>% #flatten the data
         na.omit()
rates %>% ggplot(aes(x=Date,y=value)) + facet_grid(type~asset, scales="free_y") +
  geom_line() + ggtitle("Rate Data") + theme(axis.text.x=element_text(angle = 45))
```



Even in supposedly clean and pre-checked data from Bloomberg, there were clear errors. The Norwegian Krone did not pay -60% monthly interest in the year 2000. Turkey, however, did suffer from hyper-inflation in the early 2000s, and so those high values are probably accurate. The EUR 1M Swap rate is obviously wrong, too; it's been the same value for nearly 10 years. The implied rate, which is backed out from the forward points, looks more realistic, but it can be messy.

Australia and Norway can be backfilled with the 1M swap data. More worrisome is the choppiness in the forward points (and hence the implied rate) for CHF, CZK, and SEK.

Spot Rates

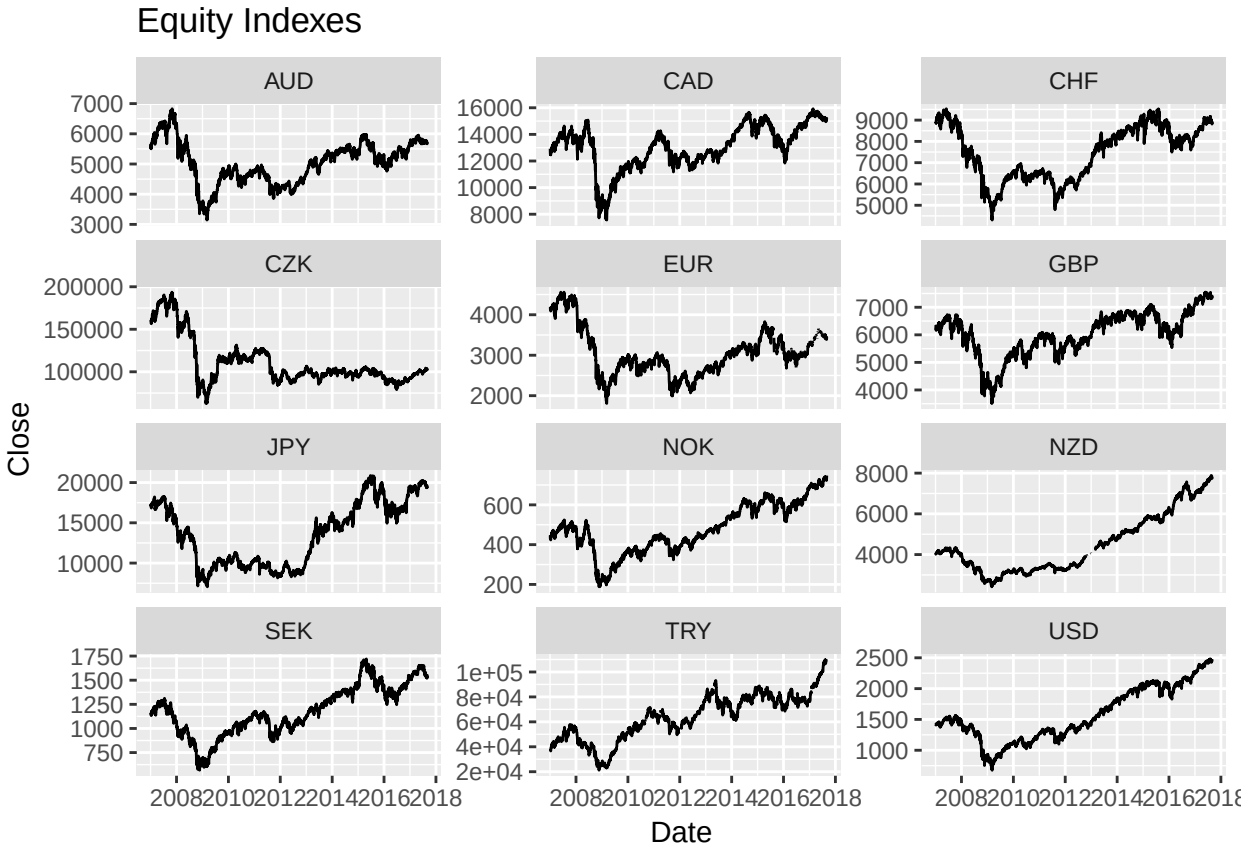
Spot rates, at least should be freely available. However, once again, data from Yahoo turned out to be poor enough that proprietary Bloomberg data was needed. Supposedly weekly data from Yahoo included some data marked from all days of the week, although not all with the same frequency. Time zone differences did not explain these errors, and so the data couldn't be trusted. There is a global standard benchmark based on prices at the close of the London exchanges, which Bloomberg provides.

Equity Indexes

These are indicators of overall stock market performance, for example the Dow and the S&P 500 indexes in the US. Data for equity indexes on Quandl was not up to date. Yahoo had most of it.

```
suppressPackageStartupMessages(library(quantmod))
stopifnot(require(xts))
EQUITIES <- c(AUD="^AXJO", CAD="^GSPTSE", CHF="SLIC.SW", CZK="FPXAA.PR", EUR="^STOXX50E",
              GBP="^FTSE", JPY="^N225", NOK="OBX.OL", NZD="^NZ50", SEK="^OMX",
              TRY="XU100.IS", USD="^GSPC")
```

```
env_eqy <- new.env() #quantmod likes to drop new variables directly into the root environment
LOADED_FROM_FILE=F
tryCatch({
  suppressMessages(suppressWarnings(
    eqy_ticks <- quantmod::getSymbols(Symbols=EQUITIES, src="yahoo",
                                      from=as.Date("2007-01-01"), env=env_eqy)))

  process_eqy <- function(ticker) {
    tick <- gsub("^\\^","",ticker)
    x <- get(tick, envir=env_eqy) #quantmod doesn't keep initial '^'s when assigning names
    colnames(x) <- gsub(paste0(tick,"\\."),"",colnames(x))
    return(data.frame(Date=index(x),coredata(x)))
  }

  map_dfr(eqy_ticks, process_eqy, .id="asset") %>%
            mutate(asset=factor(names(EQUITIES)[as.numeric(asset)]))
  },
  error=function(e) {LOADED_FROM_FILE=T;return(read_csv("data/equities.csv"))}
) -> equities
equities %>% ggplot(aes(x=Date, y=Close)) + facet_wrap(~asset, ncol=3, scale="free_y") +
  geom_line() + ggtitle("Equity Indexes")
```



Data was patchy for CHF, NZD, and TRY, and non-existant for CZK and NOK. CZK and NOK were filled in using data downloaded directly from their respective Bourse's websites.

Finally, there were some timing issues with antipodal stock exchanges. Yahoo provided all data in the US Eastern time zone, so that AUD and NZD exchanges open on Sunday night instead of Monday morning.

Since the plan is to use weekly signals, this data needed to be put back in local timezones.

```
if(!LOADED_FROM_FILE) {
equities <- equities %>% mutate(Date=ifelse(asset %in% c("AUD","NZD"),  Date + days(1),Date),
                                Date=as.Date(Date, origin=ymd(19700101)))
}
```

Factors

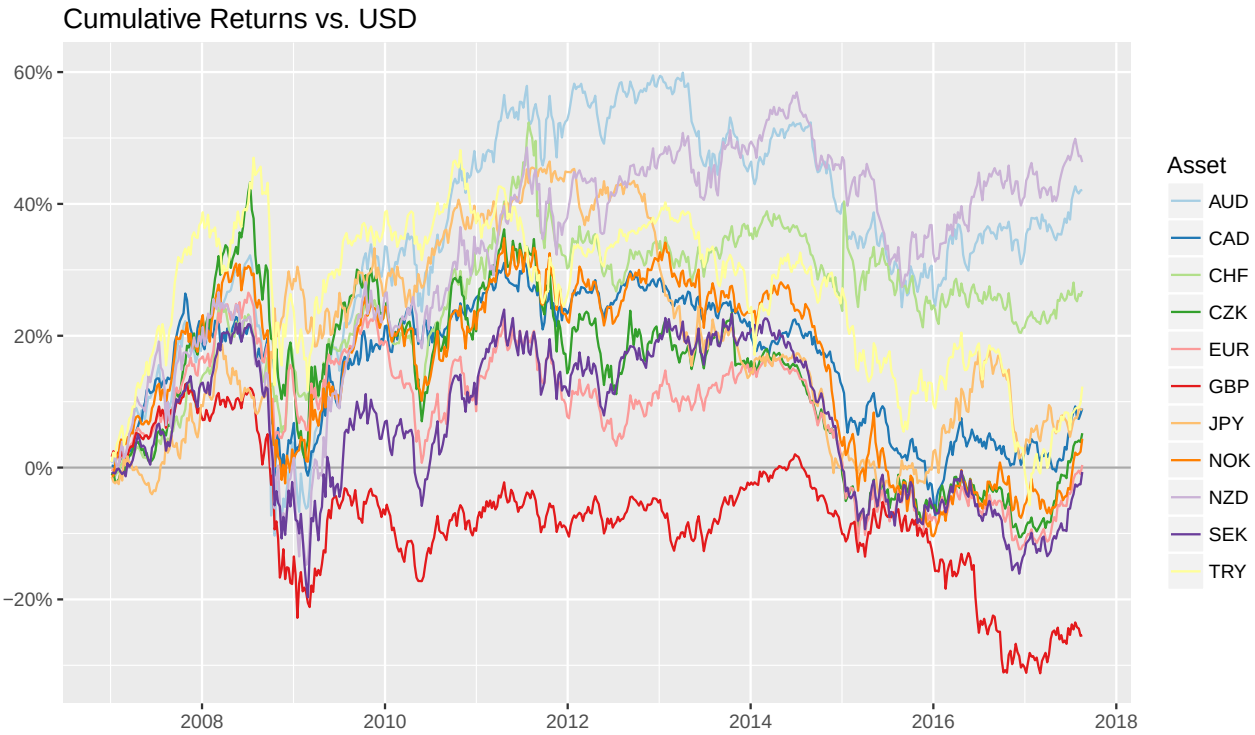
The raw data gathered above needed to be transformed into exogenous variables, typically known as 'factors' in finance, but often as 'features' in machine learning parlance. Again, due to space constraints, the full backfilling and conversion code can be found in (Naylor 2017), in the Calculating Factors section.

Y variables

The endogenous, 'Y' variable comprises weekly currency returns on 11 currencies for a ten-year period starting in 2007 and ending in 2017.

Weekly currency returns consist of the change in spot rate against the US dollar, plus the carry, defined as the (time-adjusted) difference in 1 month forward rates between the local currency and the USD.

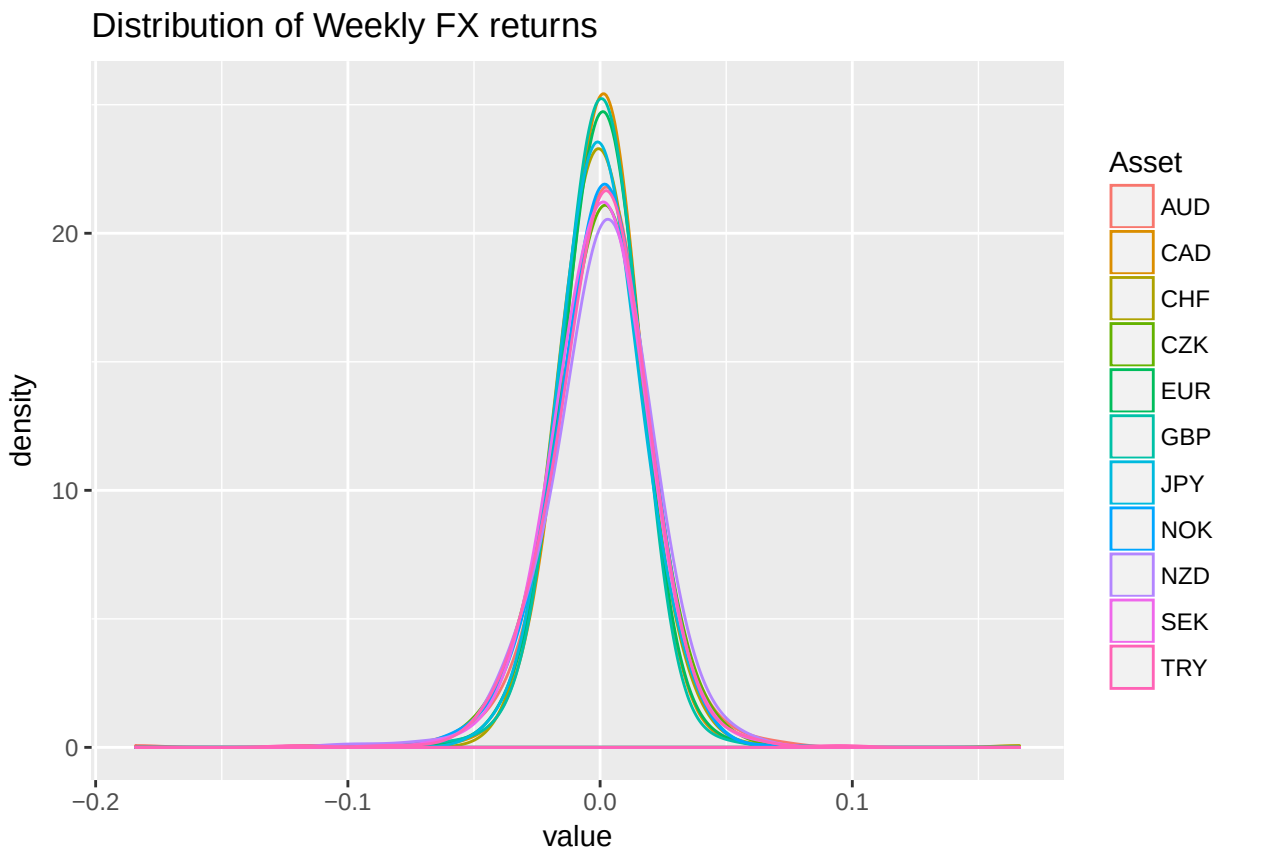
It's important to note that the endogenous variable is *multivariate*. One cannot assume that different currencies' returns are independent of one another.



The distribution of weekly asset returns shows high kurtosis, known as "fat tails" in the industry.

```
endo %>%
  gather(Asset,value,-Date) %>%
  drop_na() %>%
  ggplot(aes(x=value, col=Asset)) +
```

```
  geom_density(bw=0.01) +
  ggtitle("Distribution of Weekly FX returns")
```



X variables

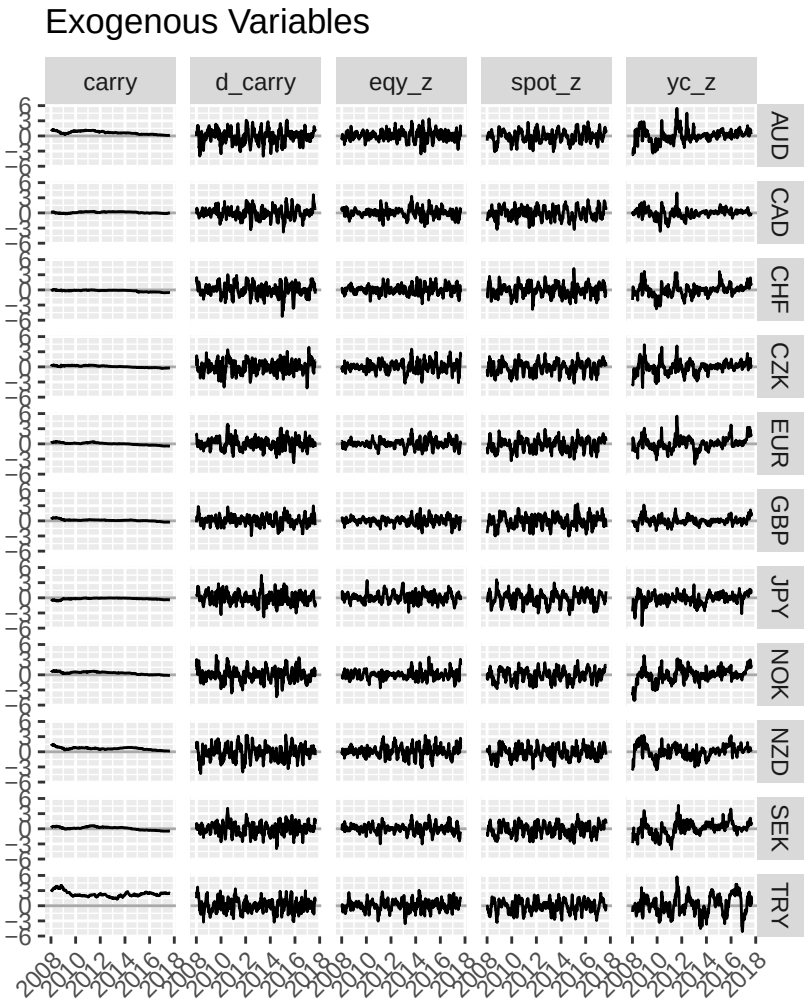
The exogenous, 'X' variables consist of weekly data for the following:

Factor	Description
Equity_d8W	The 8 week change in the local equity index
Spot_d8W	The 8 week change in spot rates
SwapRate2Y	The 2Y Swap Rate
TwoYear_d8W	The 8 week change in 2Y Swap rate
YieldCurve	The spread between 10Y and 2Y Swap rates

A great deal of currency appreciation is caused by changes in expectations of interest rates. Hence three of the five factors being used measure some aspect of those expectations. The change in Equity indexes is meant to proxy overall economic health. The change in prior spot rate is a 'momentum' indicator, a crude attempt to capture the chaotic dynamics of competitive agents in an iterative system.

There is clear room for improvement if we were to fit a model that permitted multiple time scales, such as a MIDAS regression. It would also be usual to include some sort of measure of liquidity and credit conditions.

These variables have been scaled and normalized.



Note that Carry is not a stochastic variable. It is, however, a rate, and impacts asset returns consistently at that rate. It's also worth noting the degree to which rate differentials have vanished since the Credit Crisis of 2008.

Models

Classification

Introduction: Regime Switching Models

There has been a substantial amount of work done in economic forecasting in which time periods are split up into *regimes*, periods in which markets are expected to behave similarly given similar data. For example, the markets in the run up to the bursting of the tech bubble in 2000, or the housing credit bubble in 2008, behaved substantially differently to the markets just after those bubbles. The Kalman Filter and Gaussian Process regression models attempt to account for changing reactions to market conditions by fitting a continuously changing regression to those conditions.

One alternative is a **Regime Switching Model**. Split the time periods into several regimes, then run a separate regression for each regime. The process is complicated by the fact what it can be extremely difficult to tell at any given moment in which regime markets were operating at any given time, even in hindsight.

One can attempt to determine these regimes by running various classifiers and clustering algorithms against the exogenous and endogenous data.

Dissimilarity Matrixes

First it is necessary to compute a dissimilarity matrix between periods. This calculates the distance between all of the values we have for each week.

While fitting a global set of regimes, will give a more comprehensible overview of the techniques, one set of regimes per country will be more useful for forecasting, as the primary interest is in what happens when there are differences between countries.

Global Regimes

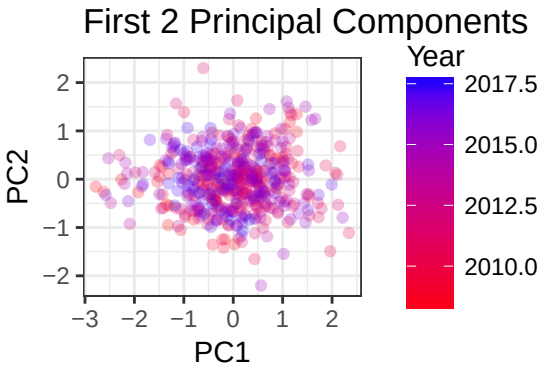
Assume the market reactions to economic conditions are the same across all countries.

```
# Organize data into per-row observation matrix
endo %>%
  gather(Asset, value, -Date) %>%
  mutate(Exog="Endo") %>%
  bind_rows(exogs) %>%
  filter(Date >=as.Date("2008-01-04")) %>% #endo has more data than exog at the beginning
  filter(Date <=as.Date("2017-08-18")) %>% #exog has more data than endo at the end
  unite(asset_exog, c("Asset", "Exog")) %>%
  spread(asset_exog, value) ->
  all_obs_matrix

all_obs_matrix %>%
  select(-Date) %>%
  daisy(metric="euclidean") ->
  global_dissimilarity
```

Principal Component Analysis

It will be hard to see what any of these clusters looks like, as the feature space is high-dimensional. First, groupings of weeks were examined using the first two principal components. These might form the basis of later visualizations.

```
all_obs_matrix %>%
  select(-Date) %>%
  prcomp() ->
  global_pca

#Plot
(global_pca$x %*% global_pca$rotation[,1:2]) %>%
  as_tibble() %>%
  bind_cols(all_obs_matrix %>%
              transmute(Year=decimal_date(Date))) %>%
  ggplot(aes(x=PC1,y=PC2, col=Year)) +
  theme_bw() +
  scale_color_gradient(low="red",high="blue")+
  geom_point(alpha=0.25) +
  ggtitle("First 2 Principal Components")
```



There is not much evidence of clear regime clusters here.

Per-Country Regimes

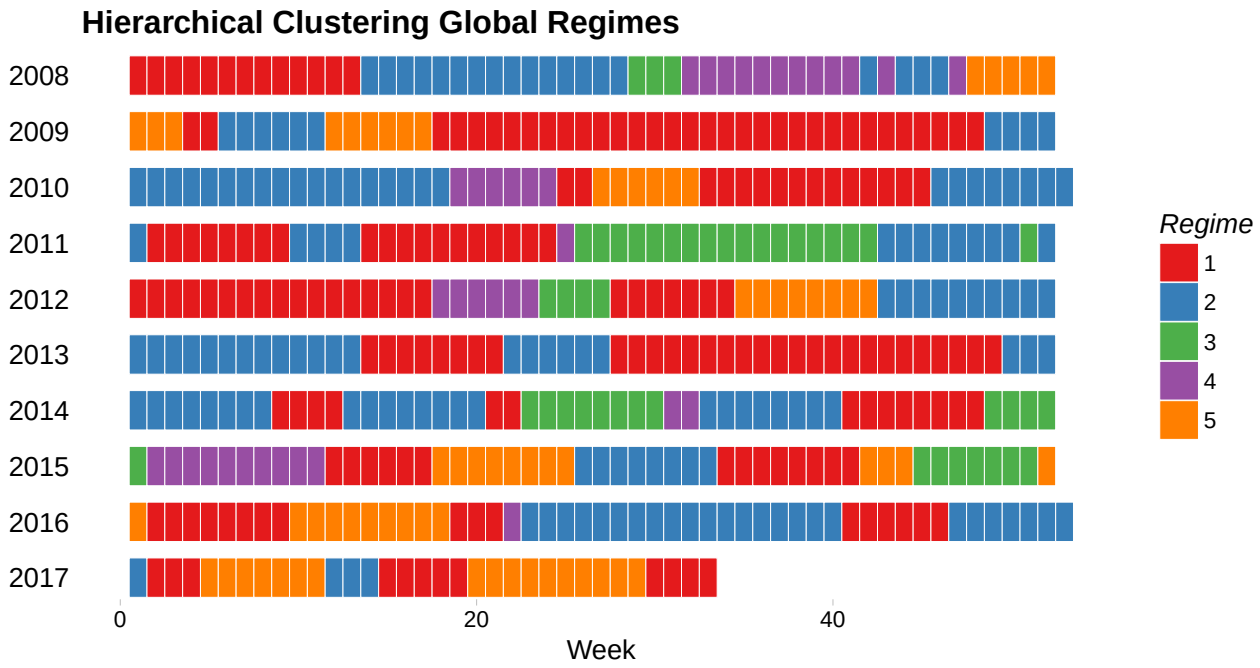
How similar are the regimes between countries? There may be an identification issue as there's no guarantee that, e.g. regime 1 in one country will be encoded as regime 1 in another, even if they refer to similar underlying clusters.

Hierarchical Clustering

In hierarchical clustering, the observations are grouped by similarity into progressively larger sections, creating a branching set of categories. Given that the observer should expect relatively few broad regimes of behavior in total, it will be necessary to cut these branches early. Probably there should only be 2 or 3 regimes in total, but for the purposes of validation, the cut was made at 5.

The identified cluster for each week can be seen below:

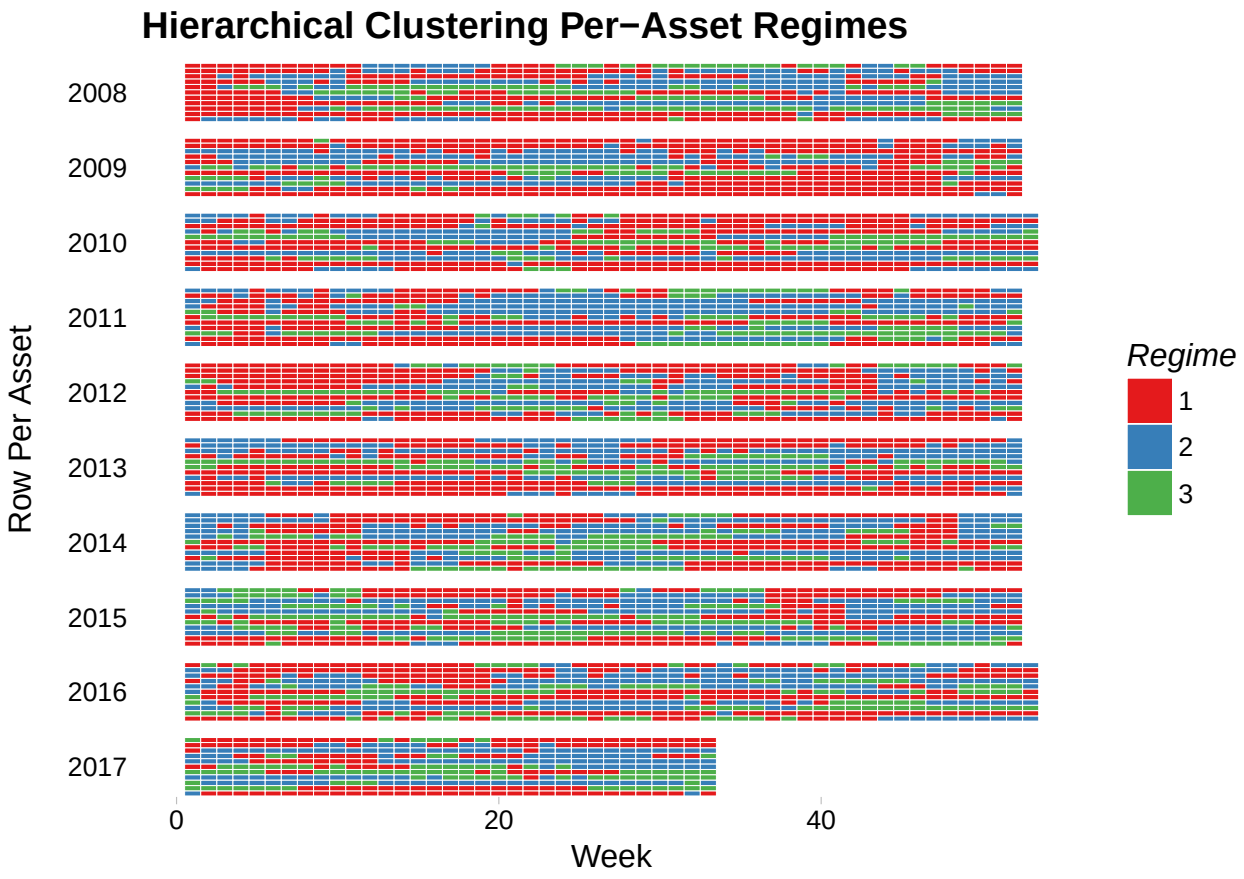
```
global_hclust <- hclust(global_dissimilarity, method='complete')

all_obs_matrix %>%
  select(Date) %>%
  mutate(Regime=factor(cutree(global_hclust, k=5)),
         Year=factor(year(Date)),
         Week=week(Date)) %>%
  ggplot(aes(x=Week, y=0, fill=Regime)) +
    theme_pander() +
    facet_grid(Year~., switch="y") +
    geom_tile(color="white") +
    scale_fill_brewer(type="qual",palette="Set1") +
    theme(strip.text.y = element_text(angle=180),
          axis.text.y=element_blank(),
          axis.ticks.y = element_blank()) + ylab("") +
    ggtitle("Hierarchical Clustering Global Regimes")
```



There are a lot of contiguous blocks, so the technique has clearly identified something.

Per-country

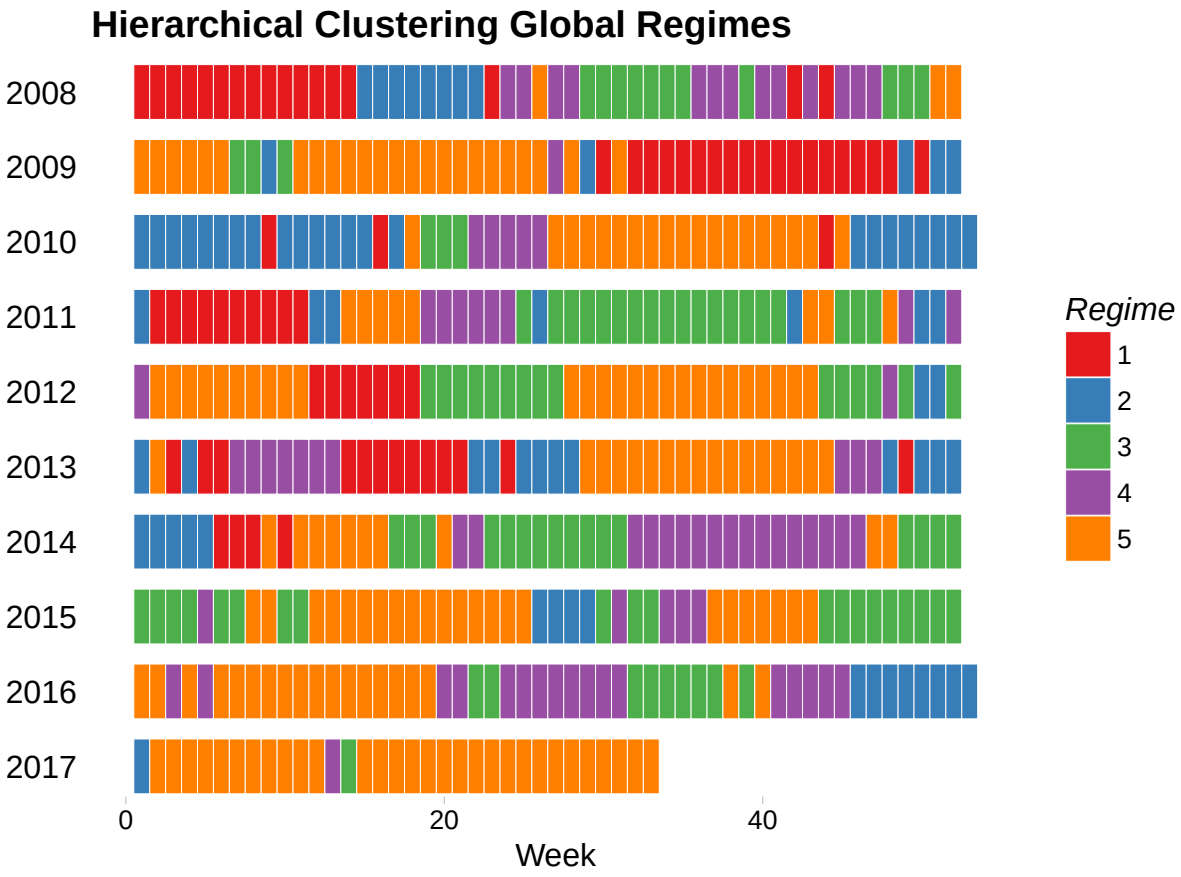
How well does the technique work when applied to individual countries?



There is still quite a lot of commonality between regimes. The least similar currency is the Turkish Lira, on the bottom row. The British Pound (middle row) enters a new regime 3 weeks *before* the vote on exiting the EU, in week 25 of 2016, which is evidence against the validity of this technique.

K-means Clustering

Unlike hierarchical clustering, K-means clustering looks at the data holistically, and attempts to create groups of points around an arbitrary number of centers. 5 centers were chosen to match the regimes identified earlier.



Although the Regime numbers are different, the pattern looks fairly similar to the global regimes made using hierarchical clustering.

Forecasting

As with most other asset classes, currency returns have fat tails, and the majority of an investor's profit (or loss) is typically made in a small number of periods. Suprise economic news or geopolitical events such as the British vote to leave the EU can cause the markets to reassess appropriate price levels drastically. As a result, a person discretizing returns so that a classifier can be applied directly would be well-advised to distinguish between large moves and smaller ones. In this analysis, discretization of the endogenous variable will be avoided as much as possible.

One must also be careful when applying typical validation techniques, particularly cross-validation, to time series. As the clustering algorithms above clearly demonstrated, one cannot take samples of the data that assume values will be independent of one another across time. The ideal equivalent to leave-one-out cross-validation is simply to run the algorithm every week with the data available for that week, keeping a single

period forecast each time. For the purposes of regressions conducted with Support Vector Machines and Random Forests, however, the data will be sliced into training and testing periods midway through the time series.

To make things convenient, move the data out of tidy format, and match up next week's returns to this week's forecasting data:

```
## Warning: Removed 3 rows containing missing values.
```

```
## Observations: 5,522
## Variables: 10
## $ Date        <date> 2008-01-04, 2008-01-04, 2008-01-04, 2008-01-04, 2...
## $ Asset       <chr> "AUD", "CAD", "CHF", "CZK", "EUR", "GBP", "JPY", "...
## $ endo        <dbl> -0.0119246426, -0.0073437496, 0.0032708987, -0.021...
## $ is_training <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TR...
## $ carry       <dbl> 0.9298046, 0.1128621, -0.1654536, 0.1353255, 0.197...
## $ constant    <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
## $ d_carry     <dbl> -0.03776817, 0.16692213, 1.53078784, 0.20935335, 1...
## $ eqy_z       <dbl> -0.29407175, 0.45187138, 0.22751153, -0.13570337, ...
## $ spot_z      <dbl> -0.94082333, -1.60569442, -0.93559429, -1.20760013...
## $ yc_z        <dbl> -3.68560665, -0.24557129, -1.99447935, -3.34662631...
```

Support Vector Machines

Although Support Vector Machines (SVMs) are traditionally used for classification, the technique can also be applied to regression.

The exogenous variables have already been standardized, so a SVM should fit reasonably well out of the box. As we are forecasting at this point, the endogenous variables (i.e. the asset returns) will need to be lagged by a week. Finally, there is the question of whether all currencies should be expected to behave similarly given the same economic data. For now, it will be assumed this is so.

Try an initial run without tuning:

```
forecast_data %>%
  filter(is_training) %>%
  select(-constant, -Date,-Asset, -is_training) %>% #a constant carries no information for a support ve
  {svm(as.formula("endo~."), data=., type="eps-regression", kernel="radial")} ->
  svm_base
summary(svm_base)
```

```
##
## Call:
## svm(formula = as.formula("endo~."), data = ., type = "eps-regression",
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.2
##     epsilon:  0.1
##
##
## Number of Support Vectors:  2504
```

2500 support vectors suggests the regression may not be robust. How well does it forecast returns for the remaining period?

```
## [1] "SSR: 0.597"
```

It remains to be seen how good that SSR score is. Practitioners typically have low expectations for goodness of fit and rely on various performance metrics after applying an optimizer to the asset forecasts to determine realistic investment decisions that could be based on the results. Asset optimization is out of scope here, however.

Tuning the SVM

The `e1071` package can perform a grid search on available parameters and tune itself, however, the algorithm assumes that cross-validation is an acceptable techniqe. The SVM will have to be tuned manually.

What about a more parsimonious representation? The SVM algorithm permits a regularization parameter, `cost`, which will penalize overly complex models. It would also be sensible to try a polynomial kernel, as asset returns, even normalized, are not expected to be linear with respect to the economic data. Finally, it is likely this regression will be noisy, so γ should be set low to reflect the high variance in the data.

```
##
## Call:
## svm(formula = as.formula("endo~."), data = ., type = "eps-regression",
##     kernel = "radial", cost = 1000)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1000
##       gamma:  0.2
##     epsilon:  0.1
##
##
## Number of Support Vectors:  2549
```

The cost parameter actually increased the number of support vectors found.

Fit a polynomial kernel:

```
##
## Call:
## svm(formula = as.formula("endo~."), data = ., type = "eps-regression",
##     kernel = "polynomial", coe0 = 0, d = 3, cost = 1000)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  polynomial
##        cost:  1000
##      degree:  3
##       gamma:  0.2
##      coef.0:  0
##     epsilon:  0.1
##
##
## Number of Support Vectors:  2495
```

This one failed to converge, so higher orders of polynomial are right out.

Finally, set gamma low:

```
##
## Call:
## svm(formula = as.formula("endo~."), data = ., type = "eps-regression",
##     kernel = "radial", cost = 1000, gamma = 0.001)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1000
##       gamma:  0.001
##     epsilon:  0.1
##
##
## Number of Support Vectors:  2514
```

Compare the four models, using Mean Squared Error this time:

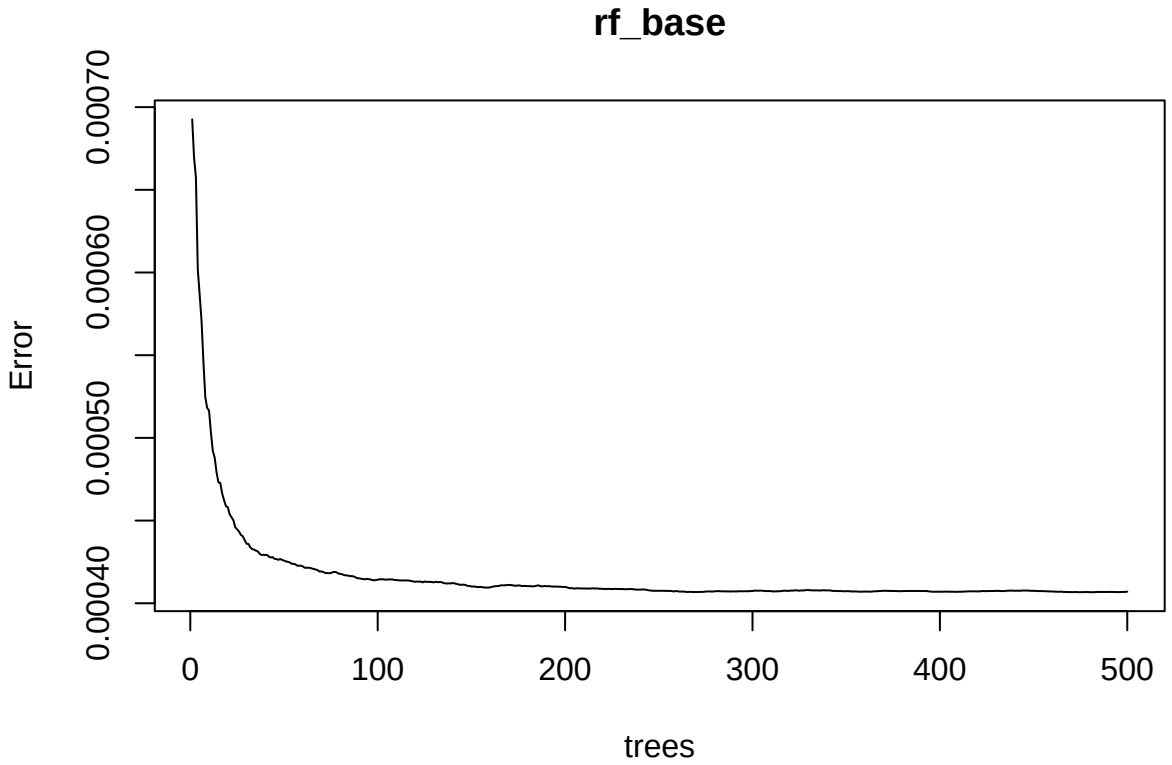
SVM	MSE
base	2.171
simple	11.369
poly	2.485
gamma	1.987

The model assuming higher noise did the best. Something very strange is going on the the high-cost 'simple' model, which wound up less simple than the base.

It's notable for all of these that the number of support vectors is close to the number of observations (around 90%), and that high penalties are not helping the model find an easier solution. SVMs are probably the wrong approach for forecasts this noisy.

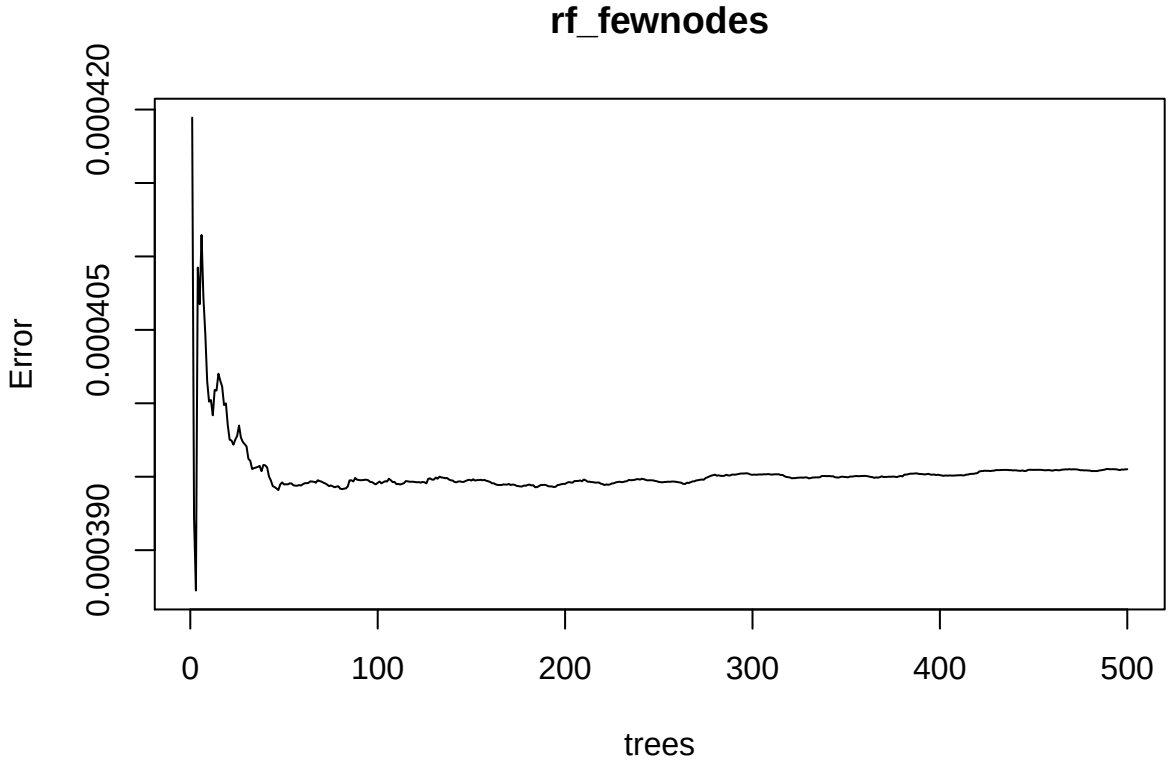
Random Forests

Decision Trees (or Regression Trees, in the continuous case) divide observations into sections according to their similarity, and relate those sections to the endogenous variable. They are notoriously unstable, however: a small change in data can result in a completely different model. Random Forests attempt to address this instability by taking random samples of the data and fitting separate sets of Regression Trees to each set, then combining the results. In their raw form, they must be used with caution as the random sampling will assume observations are independent, when in fact they are correlated over time. The `randomForest` function in R can also be set to return a proximity measure, similar to the dissimilarity matrixes calculated above for the classifiers. More on this later.



There is not much gain from fitting more than about 25 trees.

As before, there should be relatively few nodes in the tree, reflecting relatively few possible macroeconomic regimes for economies. How well does the Random Forest regression do when `maxnodes` has been set quite low?



That seems to have done just as well. Update the Mean Squared Error table:

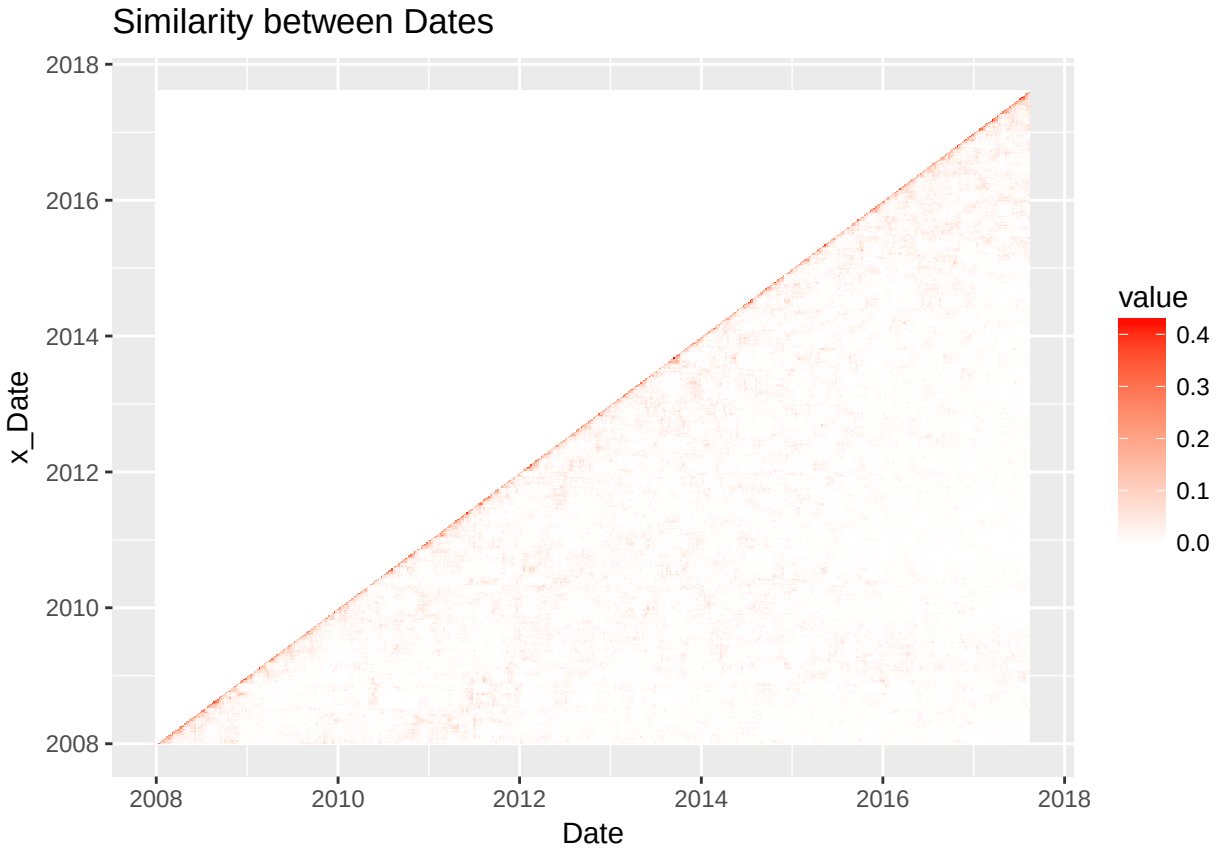
Model	MSE
svm_simple	11.369
svm_poly	2.485
svm_base	2.171
rndForest	2.124
svm_gamma	1.987
rf_fewnodes	1.956

Fewer nodes made for a more robust model, but it is only marginally better than the noise-tolerant SVM.

Analogy Weighting

Jim Savage, currently chief Data Scientist at Lendable, Inc., invented the technique of Analogy Weighting for his PhD thesis. His code is available here. One runs a weighted linear regression on time series data using the proximity matrix from a Random Forest regression for the weights. The intuition is that similar periods in history ought to elicit similar reactions to the data. Dr. Savage used the data to forecast three correlated economic time series. A key advantage of the analogy weighting methodology is that the weights can be set such that the regression is run on each data point without looking into the future. This mimics the Kalman Filter workflow in that an investor need not fit the entire model over again for each new data point. Alternately, one can see an implementation of a Gaussian Process Regression on this same data here. It was necessary to fit the model over 500 times, which took considerable processing power.

For an additional challenge, this forecast will take into account the differences between countries, while fitting a single set of betas to all. Outcomes will be modeled as arising from a multivariate normal process to mimic the manner in which different countries' currencies affect one another.



Weightings will probably need to be normalized.

Panel estimation of weighted regressions

The estimation of a multivariate endogenous variable is complex and best expressed probabilistically using a generative model. Properly, fake data should be generated with known parameters, and the model validated by showing that it can recover those parameters. An example of the validation workflow can be found can be found here for the Gaussian Process regression mentioned earlier.

The model will be fit using Stan, the current state of the art for Bayesian generative models. Stan compiles directly to C for efficiency's sake, so the code in R will be specified as a long string. Best practice with Stan is to fit using Hamiltonian Monte Carlo, but due to time constraints this project may settle for maximum likelihood optimization.

As a final input, the estimated covariance of the various currency returns will be specified directly, for parsimony's sake. A production system would typically estimate the joint volatility using a GARCH model.

```
suppressPackageStartupMessages(library(rstan))
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores()) # MCMC techniques involve running multiple chains and ensur

weighted_mvn_code <- "
data {
  int<lower=0> A; // # of Assets
  int<lower=0> T; // # of weeks (recall that the linear regression weighting permits testing on t+1 fore
  int<lower=0> J; // # of exogenous variables
  int<lower=0> T_1; // #start of training period. We need some padding so the regression has enough peri
```

```
  row_vector[A] Y[T]; // asset returns
  matrix[J,A] X[T]; //exogenous panel data
  cov_matrix[A] y_cov[T]; // Covariance of asset returns
  matrix[T,T] proxmat; //proximity matrix for weighting log-likelihood
}
transformed data {
  int<lower=0> TL; //#training period length
  matrix[A,A] L_y_cov[T];
  TL = T - T_1;
  for(t in 1:T) {
    L_y_cov[t] = cholesky_decompose(y_cov[t]); // Cholesky decompositions are more stably positive-defi
  }
}
parameters {
  matrix[TL, J] beta; // Shrunk betas across all assets. Could have been estimated as a state space mod
}
model {
  to_vector(beta) ~ normal(0, 1);
  for(t in T_1:(T-1)) { //for each week in the data
      for(i in 1:t) { //for each period prior to that week
        target += proxmat[t,i] * multi_normal_cholesky_lpdf(Y[i]|beta[t-T_1+1]*X[i], L_y_cov[i]);
      }
    }
}
generated quantities {
  //Generate forecasts so that we will have plenty of data to feed the neural net.
  vector[A] y_hat[TL];
  vector[TL] log_lik;
  log_lik[1] = multi_normal_cholesky_lpdf(Y[T_1]|beta[T_1]*X[T_1], L_y_cov[T_1]);
  for(t in (T_1+1):(T-1)) {
    log_lik[t-T_1+1] = multi_normal_cholesky_lpdf(Y[t]|beta[t-T_1+1]*X[t-T_1+1], L_y_cov[t-T_1+1]);
    y_hat[t-T_1+1] = multi_normal_cholesky_rng(to_vector(beta[t-T_1]*X[t-T_1+1]), L_y_cov[t-T_1+1]);
  }
}
"

weighted_mvn_model <- stan_model(model_name="weighted_mvn", model_code=weighted_mvn_code)
```

```
## In file included from /home/cnaylor/R/x86_64-pc-linux-gnu-library/3.2/BH/include/boost/config.hpp:39:0
##             from /home/cnaylor/R/x86_64-pc-linux-gnu-library/3.2/BH/include/boost/math/tools/config.hp
##             from /home/cnaylor/R/x86_64-pc-linux-gnu-library/3.2/StanHeaders/include/stan/math/rev/cor
##             from /home/cnaylor/R/x86_64-pc-linux-gnu-library/3.2/StanHeaders/include/stan/math/rev/cor
##             from /home/cnaylor/R/x86_64-pc-linux-gnu-library/3.2/StanHeaders/include/stan/math/rev/cor
##             from /home/cnaylor/R/x86_64-pc-linux-gnu-library/3.2/StanHeaders/include/stan/math/rev/mat
##             from /home/cnaylor/R/x86_64-pc-linux-gnu-library/3.2/StanHeaders/include/stan/math.hpp:4,
##             from /home/cnaylor/R/x86_64-pc-linux-gnu-library/3.2/StanHeaders/include/src/stan/model/mo
##                  from file4e5f3861f562.cpp:8:
## /home/cnaylor/R/x86_64-pc-linux-gnu-library/3.2/BH/include/boost/config/compiler/gcc.hpp:186:0: warni
##  #  define BOOST_NO_CXX11_RVALUE_REFERENCES
##  ^
## <command-line>:0:0: note: this is the location of the previous definition
```

Prepare the data in the format Stan expects. This was defined in the `data` section above.

Σ_y

Create exponentially weighted variance-covariance matrix. GARCH would be better, but from prior experience makes little difference to weekly currency returns.

```
FIRST_DATA_SET <- 52 #Start the forecast after we have at least 52 weeks of data
N1 <- nrow(endo) - 1 - FIRST_DATA_SET
N2 <- 0 # # of extra periods
A <- ncol(endo)-1

y_cov <- array(NA, dim=c(N1+N2, A, A))

lambda <- exp(log(0.5)/52) # 1Y half-life exponential decay
wts <- lambda ^ (seq.int(0, y_offset+N1+N2))
for(n in 1:nrow(y_cov)) {
  ind <- seq.int(FIRST_DATA_SET+n-1) #extra -1 in the index because we are lagging these an extra step
  y_cov[n,,] <- cov.wt(endo[ind,-1],rev(wts[ind]), center=F)$cov
}
```

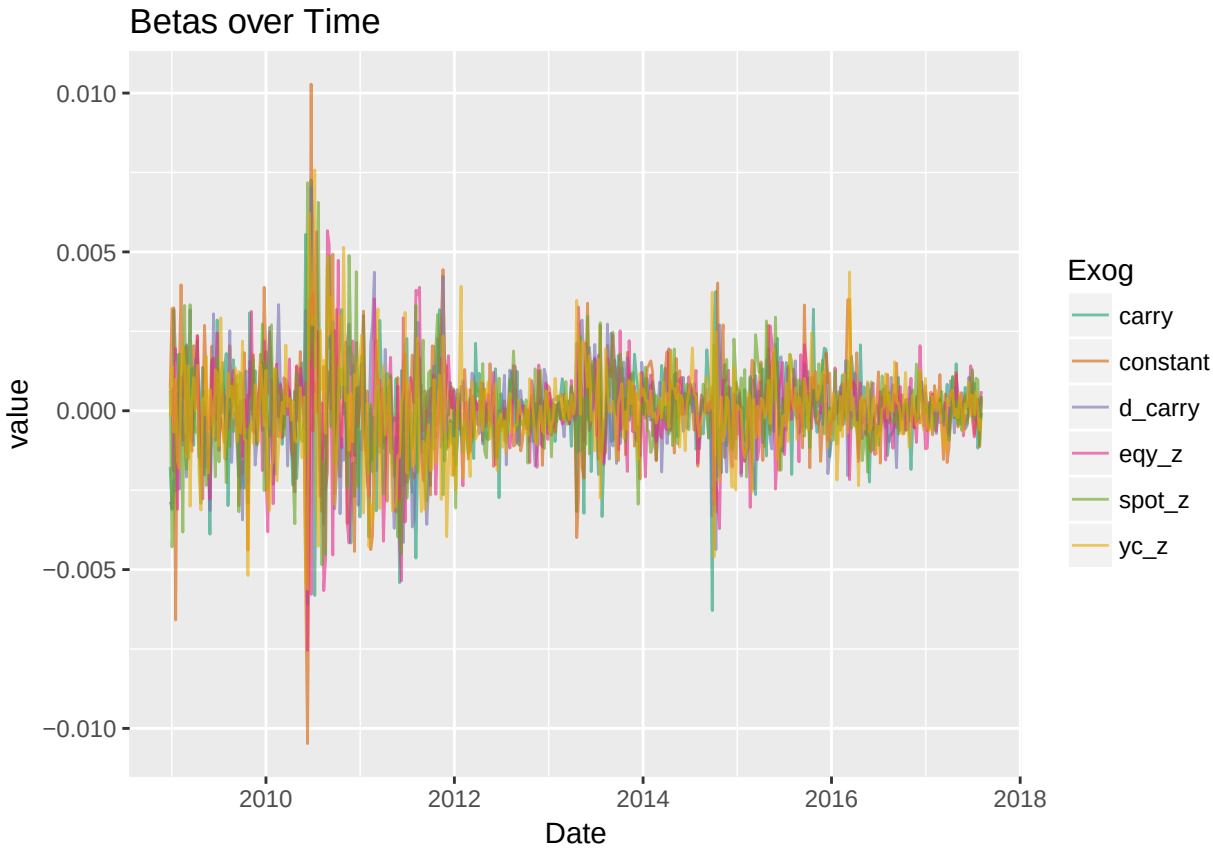
Prepare the rest of the data

```
stan_data <- list(A=length(unique(forecast_data$Asset)),
                  T=length(unique(forecast_data$Date)),
                  J=6,T_1=52,
                  Y=forecast_data %>%
                    select(Date, Asset, endo) %>%
                    spread(Asset, endo) %>%
                    select(-Date),
                  X=forecast_data %>%
                    select(-endo, -is_training, -Date) %>%
                    nest(-Asset) %>%
                    {abind(.$data, along=3, new.names=.$Asset)},
                  y_cov=y_cov[3:504,,],
                  proxmat=proximity %>%
                    ungroup() %>%
                    spread(x_Date, value) %>%
                    select(-Date) %>%
                    as.matrix())
```

Run direct maximum likelihood optimization instead of Hamiltonian Monte Carlo due to time constraints.

```
fit_weighted_mvn <- optimizing(weighted_mvn_model, data=stan_data)
```

```
## Initial log joint probability = -4.2079e+08
## Optimization terminated normally:
##   Convergence detected: relative gradient magnitude is below tolerance
```



The betas could be artificially induced to be more stable, as in a State Space Model. The fact that they are not naturally so, and have such small values, indicates that most of the variation in asset returns cannot be explained by the weighted model.

How well did the model predict returns?

Model	MSE
svm_simple	11.369
analogy_mvn	5.850
svm_poly	2.485
svm_base	2.171
rndForest	2.124
svm_gamma	1.987
rf_fewnodes	1.956

On its own, the analogy-weighted model is one of the poorest options.

Taming the Chaos

The introduction mentioned the chaotic nature of currency returns. Recently, a team at the University of Maryland have had some success in forecasting chaotic time series several steps ahead (Pathak et al. 2018). They took a deterministic chaotic system known as the Kuramoto-Sivashinksy equation, and fed measurements into a set of 64 Echo State Network layers, with each layer handling a smaller section of the overall data. An Echo State Network layer consists of a relatively large number of nodes with sparse, random connections (essentially an Erdos-Renyi network rather than the scale-free network believed to be the actual

characterization of currency returns) (Lukoševičius 2012). The input for this implementation will consist of the residuals from the Analogy-weighted regression computed before.

Unfortunately, Echo State Networks have fallen out of fashion in recent years after the success of LSTM networks. The author was unable to implement an Echo State Network cell manually in tensorflow or keras. Instead, a standard LSTM network will stand in on the last stage of forecasting. A few designs were tried but the best results came from two LSTM layers fit with a relatively high 50% Dropout layer interspersed between them. The high Dropout was an effort to mimic the sparsely linked nodes typical of Echo State Networks, The LSTM node values were converted back to relevant residuals using a set of linear weights. While the model should be fittable using R's interface to Python, issues in data structure translation led the code to be run directly in Python. It can be found in the accompanying file, `LSTM.ipynb`.

Add the predicted residuals to the predicted Y values from the Analogy weighting, and a new, hybrid Bayesian/Deep Learning model arises. Albeit on a shorter time horizon, as the LSTM was given two years on which to train.

Syracuse's SUrge GPU cluster helpfully provided the hardware needed to run the network.

```
read_csv("data/lstm_resids.csv",
         col_types=cols(Date=col_date(),
                        .default=col_double())) %>%
  gather(Asset, r_hat, -Date) %>%
  inner_join(y_hat, by=c("Asset", "Date")) %>%
  inner_join(endo %>% gather(Asset, y, -Date), by=c("Asset","Date")) %>%
  summarize(Model="analogy+lstm", MSE=10000*mean((y-(y_hat+r_hat))^2)) %>%
  bind_rows(MSE) %>%
  arrange(desc(MSE)) ->
  MSE
MSE %>% knitr::kable(digits=3)
```

Model	MSE
svm_simple	11.369
analogy+lstm	6.202
analogy_mvn	5.850
svm_poly	2.485
svm_base	2.171
rndForest	2.124
svm_gamma	1.987
rf_fewnodes	1.956

Unfortunately, the LSTM network failed to predict the residuals with any accuracy. Perhaps a different configuration, a proper Echo State Network, or a great deal more data would make the difference.

Conclusions

Of the data mining techniques tried, Random Forests and fault-tolerant Support Vector Machines showed the most promise, although both algorithms looked to have formed specifically fitted models, which may not perform as well for new data.

The classifiers showed promise, producing contiguous and generally sensible blocks of similar behavior. There were a few specific events, however, that should have signaled shifts in regime and did not. Particularly, the hierarchical clustering algorithm showed a shift in the British Pound's behavior 3 weeks *before* the vote to

leave the EU took place, when in fact sharp moves in the spot rate occurred that night in the Asian markets as the per-county results came in.

Efforts to use the probabilistic classification techniques afforded by the proximity matrix generated by the random forest algorithm to weight a Least Squares regression were not successful, although more work could be done on the subject. An obvious addition would be to smooth out the feature betas using a state space model.

A great deal more work remains to be done on layering Deep Learning or Reservoir Computing techniques on top of traditional regressions, as suggested in (Pathak et al. 2018). Thus far, however, the hybrid model has not been a success. Perhaps better results could be had if the regression were also fit within the Tensorflow model. Another alternative might be to compensate for the lack of data by using the generative output from an MCMC regression instead of the point estimates of asset returns.

References

Barabasi, Albert-Laszlo. 2002. "Linked the New Science of Networks." Cambridge, Mass.: Perseus Pub. http://www.amazon.com/Linked-The-New-Science-Networks/dp/0738206679.

Lukoševičius, Mantas. 2012. "A Practical Guide to Applying Echo State Networks." In *Neural Networks: Tricks of the Trade: Second Edition*, edited by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller, 659–86. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-35289-8_36.

Naylor, Charles. 2017. "Gaussian Process Regression for Fx Forecasting: A Case Study." https://charlesnaylor.github.io/gp_regression/.

Pathak, Jaideep, Brian Hunt, Michelle Girvan, Zhixin Lu, and Edward Ott. 2018. "Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach." *Phys. Rev. Lett.* 120 (2). American Physical Society: 024102. doi:10.1103/PhysRevLett.120.024102.