Financial Data Project: Notes on Factor Analysis

Dom Owens

18/11/2019

Factor Analysis

Within this project, we will use **Factor Analysis** as a dimension reduction technique for our multivariate time series

$$\mathbf{x}_t$$

The problem is treated as follows:

We wish to reduce the dimension of an observable random vector

$$\mathbf{x} \in \mathbb{R}^p$$

to a smaller vector

$$\mathbf{x} \in \mathbb{R}^m$$

of latent variabes, where $m << p$.

We do this by expressing each entry x_i as a linear combination of the factors:

$$x_i = \lambda_{i1} f_1 + ... + \lambda_{im} f_m + \epsilon_i \qquad \mathbf{x} = \Lambda \mathbf{f} + \boldsymbol{\epsilon}$$

Here, Λ are the **factor loadings**, and

$$\boldsymbol{\epsilon}$$

are errors.

We make the following assumptions:

- $E(\boldsymbol{\epsilon}) = \mathbf{0}$, $E(\boldsymbol{f}) = \mathbf{0}$, $E(\boldsymbol{x}) = \mathbf{0}$ (WLOG)
- $E(\boldsymbol{\epsilon}\boldsymbol{\epsilon}^{\boldsymbol{T}}) = \boldsymbol{\Psi}$ is a diagonal matrix
- $E(\boldsymbol{f}\boldsymbol{f}^{\boldsymbol{T}}) = \boldsymbol{I}_m$, so that the factors are independent
- For inferential purposes, we make distributional assumptions on $\boldsymbol{f}$ or $\boldsymbol{x}$ (often multivariate normality)

We will be working with **time series** data and models, meaning our observations

$$\boldsymbol{x}_t$$

are indexed in time by $t \in \{0, 1, ...T\}$. We further assume that

- The **Covariance** matrix Σ is constant with respect to t (this is the **static** model, as opposed to the more complicated **dynamic** model)
- The differenced logarithm of the series is second-order stationary

Indeed, underlying the whole idea of forecasting financial markets is the *Big Assumption*, which is that economic activity in the near future will closely resemble economic activity in the past.

Estimation

As opposed to the similar-looking regression problem

$$\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

in which the desgin matrix X is known, we know neither $\boldsymbol{f}$ nor Λ; hence, any "best-fit" solutions will not be unique.

For conducting estimation in practice, we often find $\hat{\Lambda}$ and $\hat{\Phi}$, then find $\hat{\boldsymbol{f}}$. In disciplines such as finance and economics, the factors can be pre-specified according to theoretical justifications (see the Fama-French factor model; we will use a mathematical approach instead.

One way of finding estimates is through Principal Components Analysis (PCA), called **Principal Factor Analysis (PFA)**. is what we will use in this project.

We work with the covariance and sample covariances matrices

$$\Sigma = \Lambda\Lambda^T + \Psi \text{ and } S = \frac{1}{T}XX^T$$

Suppose that we have the principal components decomposition

$$\boldsymbol{x}_t = A^T\boldsymbol{z}_t$$

where $A \in \mathbb{R}^{p\times n}$ is a matrix consisting of eigenvectors $\boldsymbol{\alpha_i}$, each corresponding to an eigenvalue l_i in decreasing order.

$$\boldsymbol{z} \in \mathbb{R}^p$$

where \$ p \$ is the number of different series being measured.

By partitioning the decomposition into the principal m and minor $p-m$ components, we obtain a factor analysis and error accordingly:

$$\begin{aligned}\mathbf{x}_t &= (A_m|A^*_{p-m})^T\begin{pmatrix}\mathbf{z}_{m,t}\\ \mathbf{z}^*_{p-m,t}\end{pmatrix}\\ &= A_m^T\mathbf{z}_{m,t} + (A^*_{p-m})^T\mathbf{z}^*_{p-m,t}\\ &= \Lambda\boldsymbol{f}_t + \boldsymbol{\epsilon}_t\end{aligned}$$

Substituting in the sample covariance and normalising gives us our estimates (for a model with $\Psi = \sigma^2 I$ this maximises the log-likelihood function (PRML p.548); here we do not assume this)

$$\hat{\Lambda} = \sqrt{p}A_m^T, \hat{\Phi} = pA^{*T}_{p-m}A^*_{p-m}$$

Factor Analysis: Selecting the number of factors m

We can select the number of factors to use, m, using the knowledge that each factor explains a decreasing amount of the total variance, given our static model estimated with PCA.

We might plot the Scree plot (with `screeplot`) and identify a drop-off in the variance explained, or pick an amount of factors sufficient to explain, say, 70% of the variance. Alternatively, information criteria can give a systematic means of selecting m, though these are slightly more complicated.

Forecasting

We opt for the **direct forecast** procedure, where data at $t + h$ is projected onto factors $\mathbf{f}_t$.

Forecasting can be conducted for a single series y_t, which is dependent on the factors, or for the whole vector of series $\mathbf{x}_t$.

References

Pattern Recognition and Machine Learning, Christopher Bishop

Dynamic Factor Models Matteo Barigozzi

Forecasting Using Principal Components From a Large Number of Predictors Stock, Watson

Financial Data Project: Forecasting Oil Movements with Factors

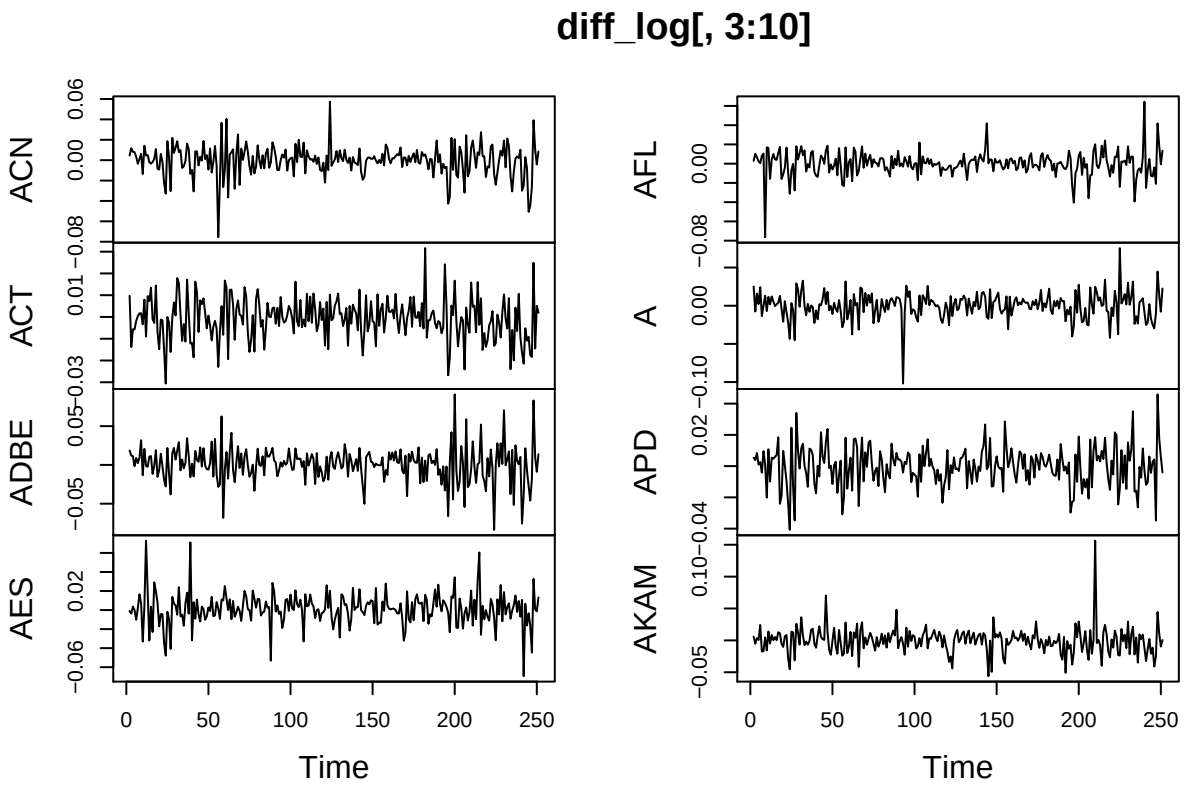
Dom Owens

03/12/2019

```
oil_close_data <- read.csv("oil_close_2018.csv") #import data
```

For financial series, we consider the logarithm of the value, since movements tend to occur multiplicitavely. These do not appear to be stationary, so we take differences.

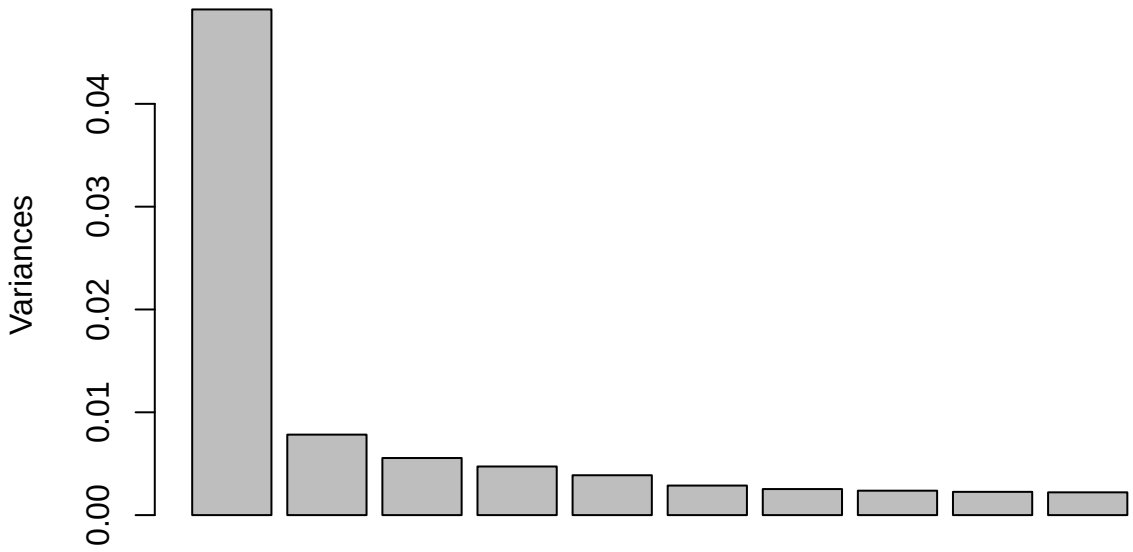
```
close_data_2018 <- subset(oil_close_data, select = -c(X.1, DATE, DCOILWTICO, ADT))
#drop date and oil, ADT
log_close <- log(close_data_2018) #take log
diff_log <- diff(ts(log_close)) #diff series
plot(diff_log[, 3:10]) #plot first 8 series
```



Extracting Factors with Principal Components

We want to find the principal components of the covariance matrix of the series. We do this using the sample covariance, the default option given by `prcomp`.

```
#Covar <- cov(diff_log) #find covariance
#PCA <- princomp(Covar) #take PCA of covariance
PCA <- prcomp(diff_log) #find PCs of sample covariance
screeplot(PCA) #plot in decreasing order of variance
```

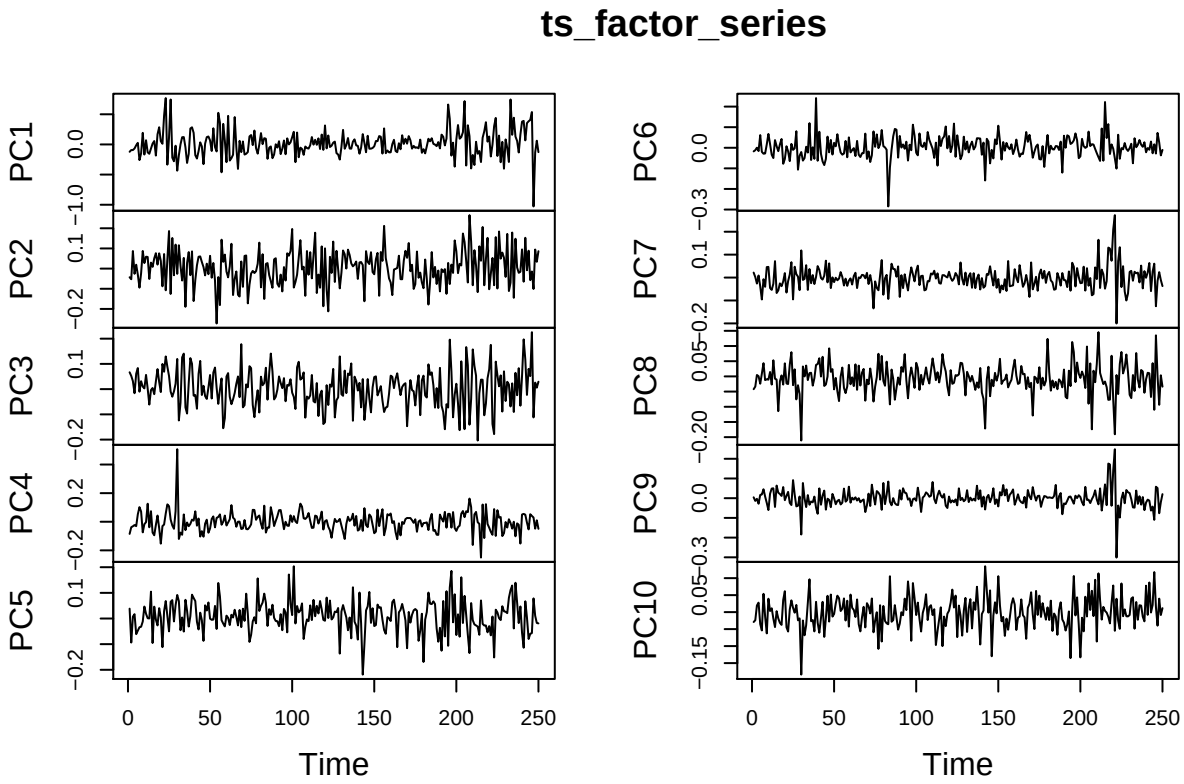



```
head(PCA$sdev, 30) #view standard deviations of first 30 components
```

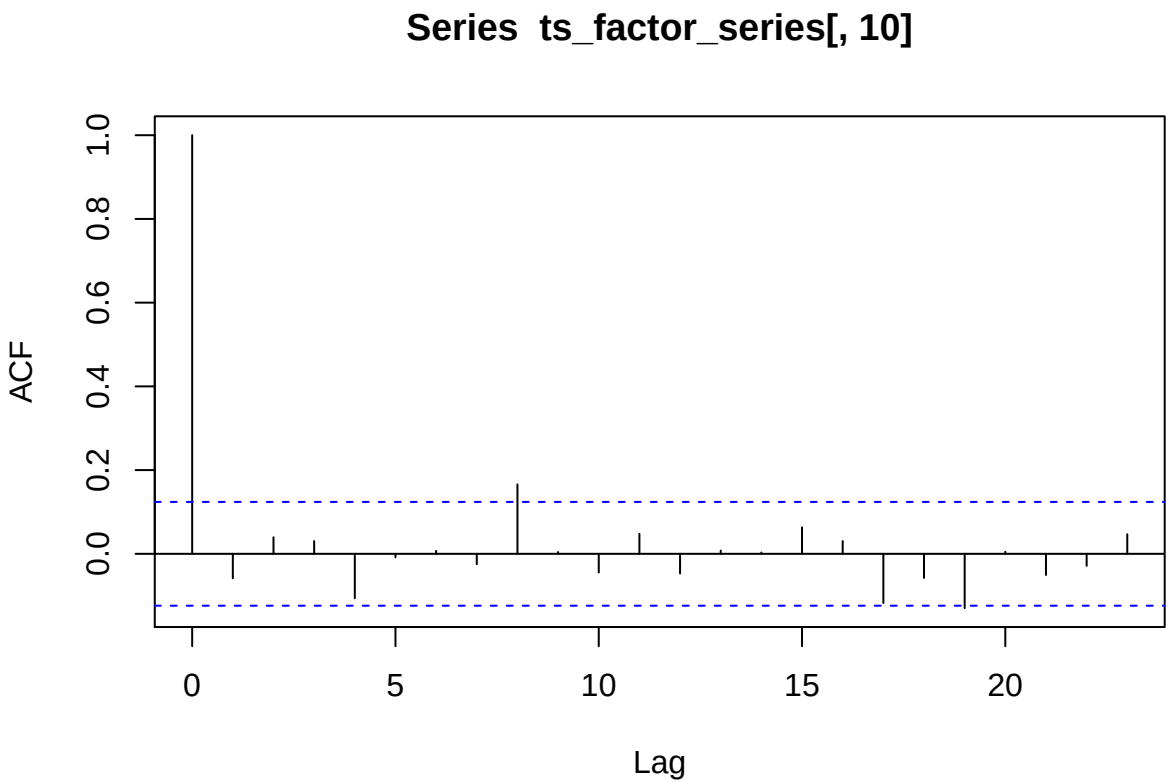
```
##  [1] 0.22176908 0.08847104 0.07451144 0.06877110 0.06226909 0.05361267
##  [7] 0.05034662 0.04874833 0.04756765 0.04706843 0.04440906 0.04257630
## [13] 0.04116579 0.04015350 0.03917405 0.03825732 0.03715776 0.03645910
## [19] 0.03583580 0.03514537 0.03488699 0.03442400 0.03424505 0.03375637
## [25] 0.03297181 0.03235025 0.03180969 0.03145715 0.03120196 0.03091355
```

We arbitrarily select the first 10 components.

```
Y <- matrix(diff_log, nrow = dim(diff_log)[1], ncol = dim(diff_log)[2] )
loadings <- PCA$rotation[,1:10] #extract first 10 components
loadings <- as.matrix(loadings)
factor_series <- (Y%*%loadings) %*% solve(t(loadings)%*%loadings) #compute time series of factors, unno

ts_factor_series <- ts(factor_series) #isolate as time series
plot(ts_factor_series) #plot factors over time
```



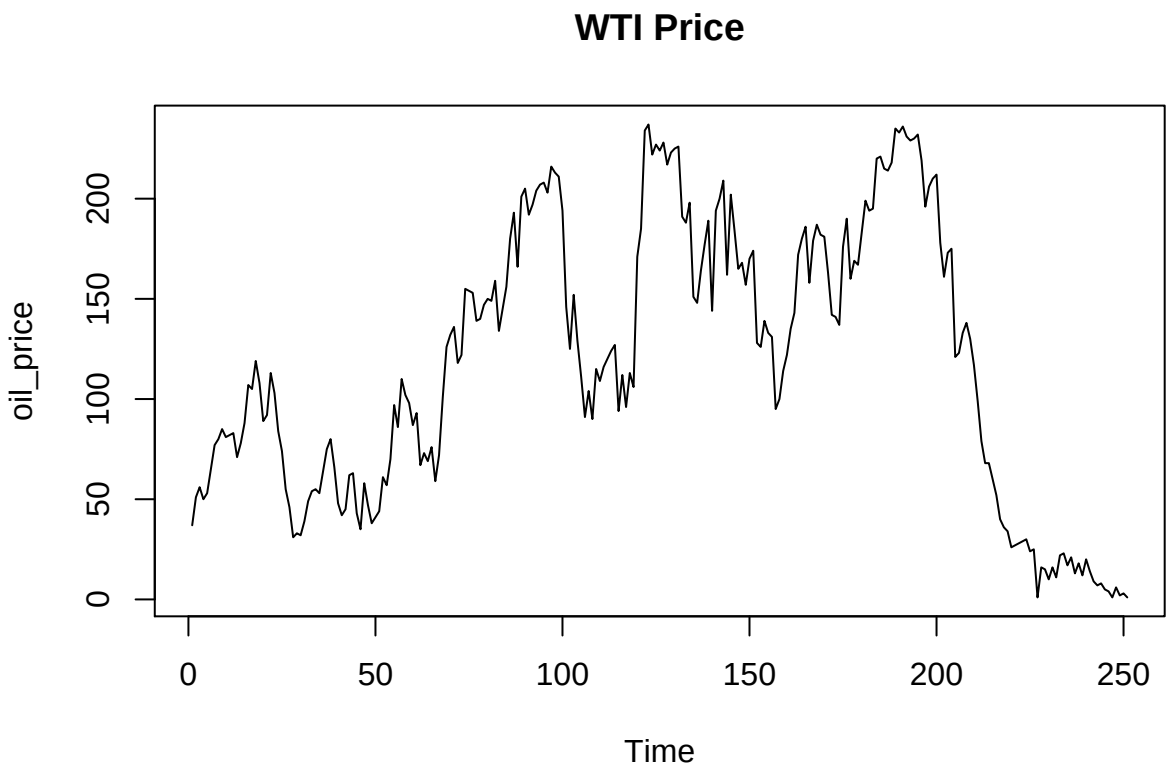

```
acf(ts_factor_series[,10]) #plot autocorellations
```



Forecasting oil prices using factors

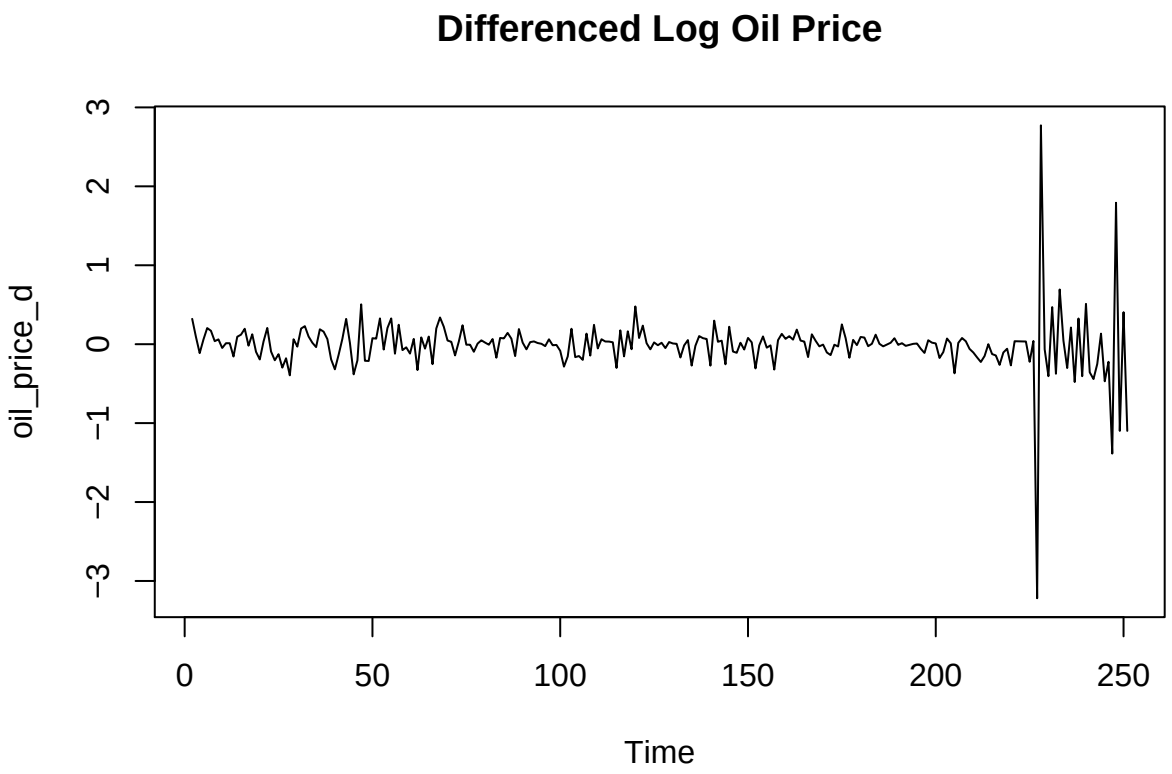
Suppose we wish to forecast the spot price of a different, but related, asset. We might think there is some relationship between the price of US-produced crude oil WTI and the S&P 500; we construct a regression on our factor representation to describe this.

```
oil_price <- ts(oil_close_data$DCOILWTICO) #select prices
plot(oil_price, main = "WTI Price")
```



Again, this is not stationary and is financial, so we take the log and difference. We assume this is stationary.

```
oil_price_l <- log(oil_price) #log series
oil_price_d <- diff(oil_price_l) #difference log series
plot(oil_price_d, main = "Differenced Log Oil Price")
```



We model the differenced log-oil price y_t as a linear function of our factors $\mathbf{f}_t$:

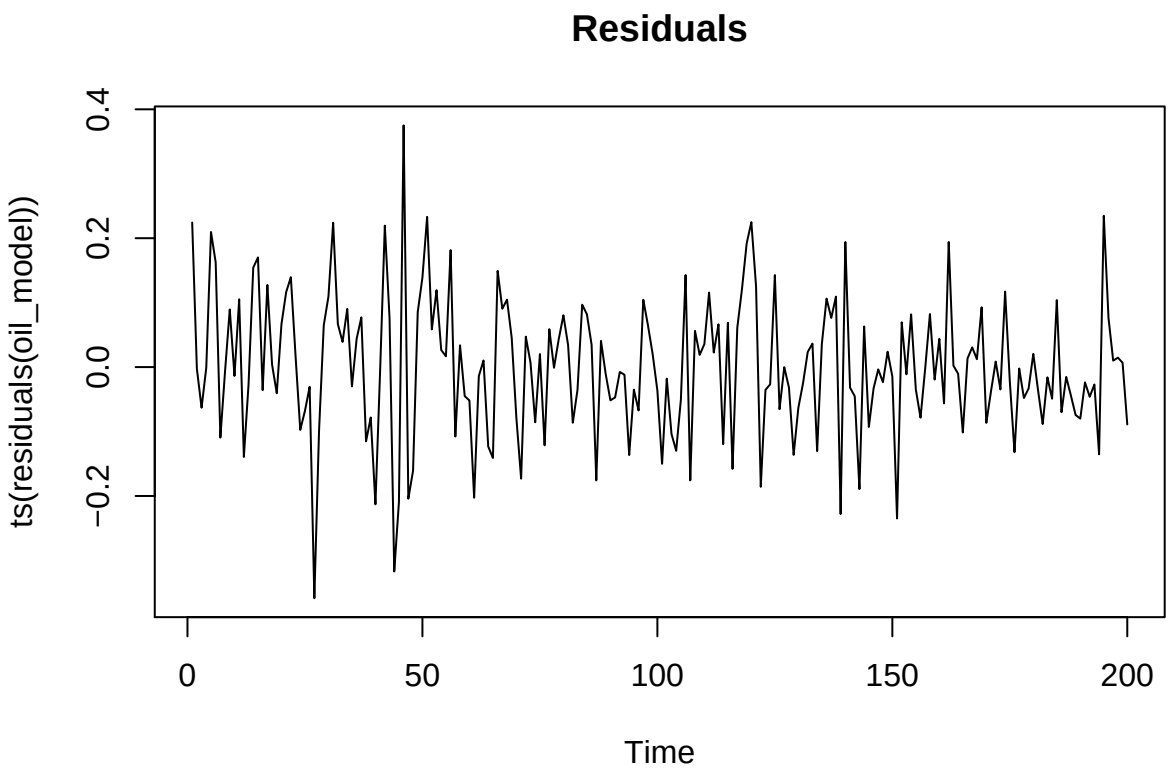
$$y_t = \boldsymbol{\beta}^T \mathbf{f}_t + \epsilon_t$$

where $\boldsymbol{\beta}$ are regression coefficients, and $\boldsymbol{\epsilon}_t$ are IID errors.

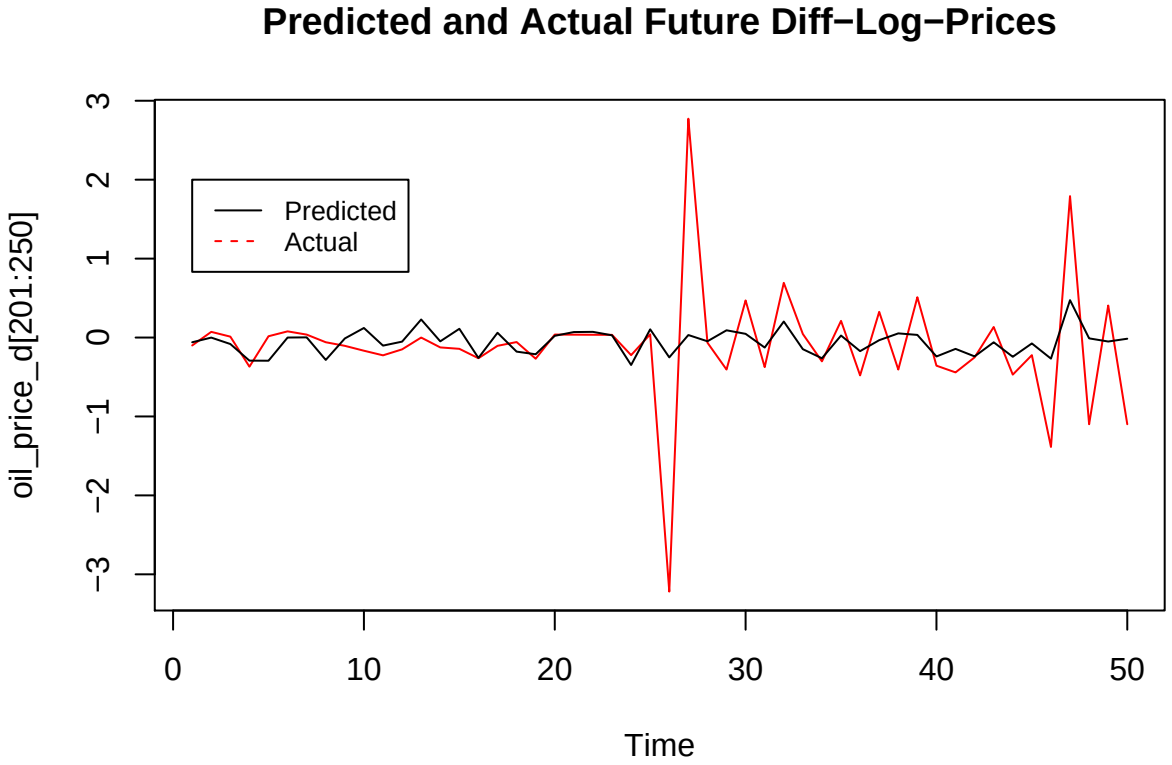
```
model_data <- data.frame(oil = oil_price_d, factor_series )#group series as dataframe
train_data <- model_data[1:200,] #select first 200 as training
test_data <- model_data[201:250,] #select last 250 as testing

oil_model <- lm(oil ~ ., data = train_data, na.action = NULL) #fit linear model

mean(residuals(oil_model)^2) #Mean Squared Error
```

```
## [1] 0.01171531
```

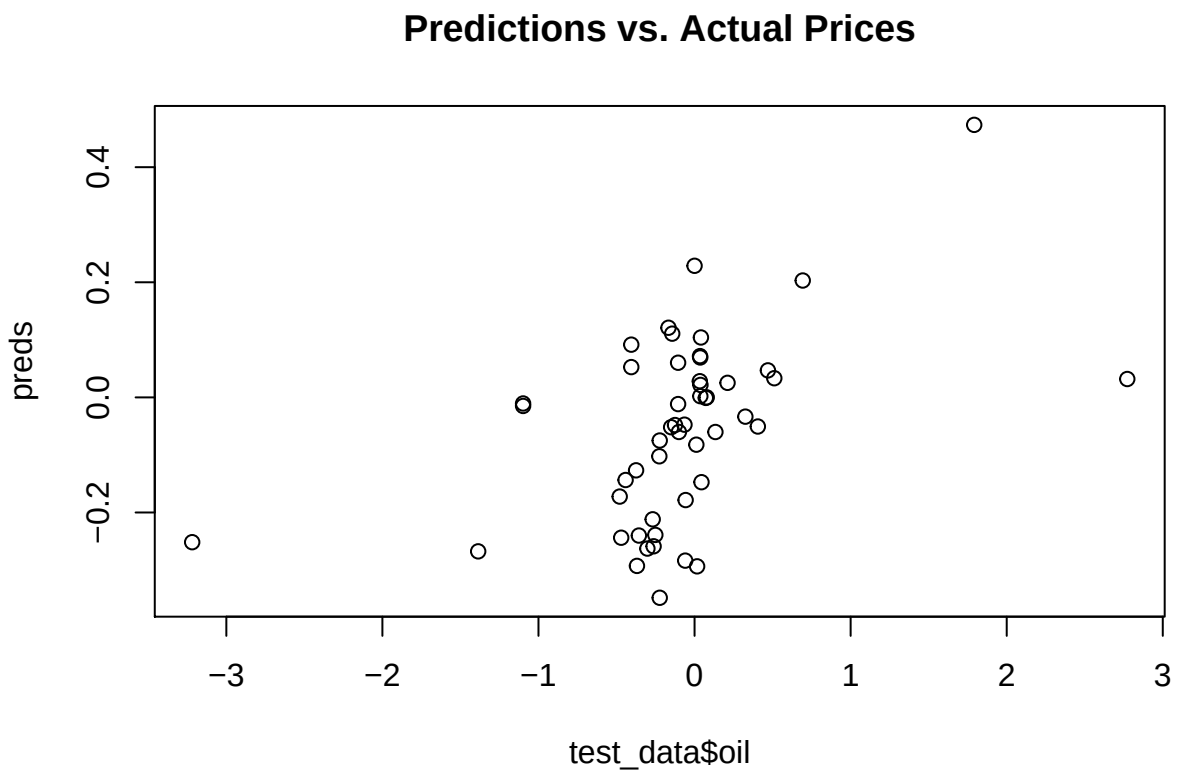
```
plot(ts(residuals(oil_model)), main = "Residuals") #residual plot
```




```
preds <- predict(oil_model,  newdata = test_data)
ts.plot(oil_price_d[201:250], col = "red", main = "Predicted and Actual Future Diff-Log-Prices") #plot
lines(ts(preds)) #overlay observed series
legend(1, 2,  legend=c("Predicted", "Actual"),
       col=c("black", "red"), lty=1:2, cex=0.8) #add legend
```



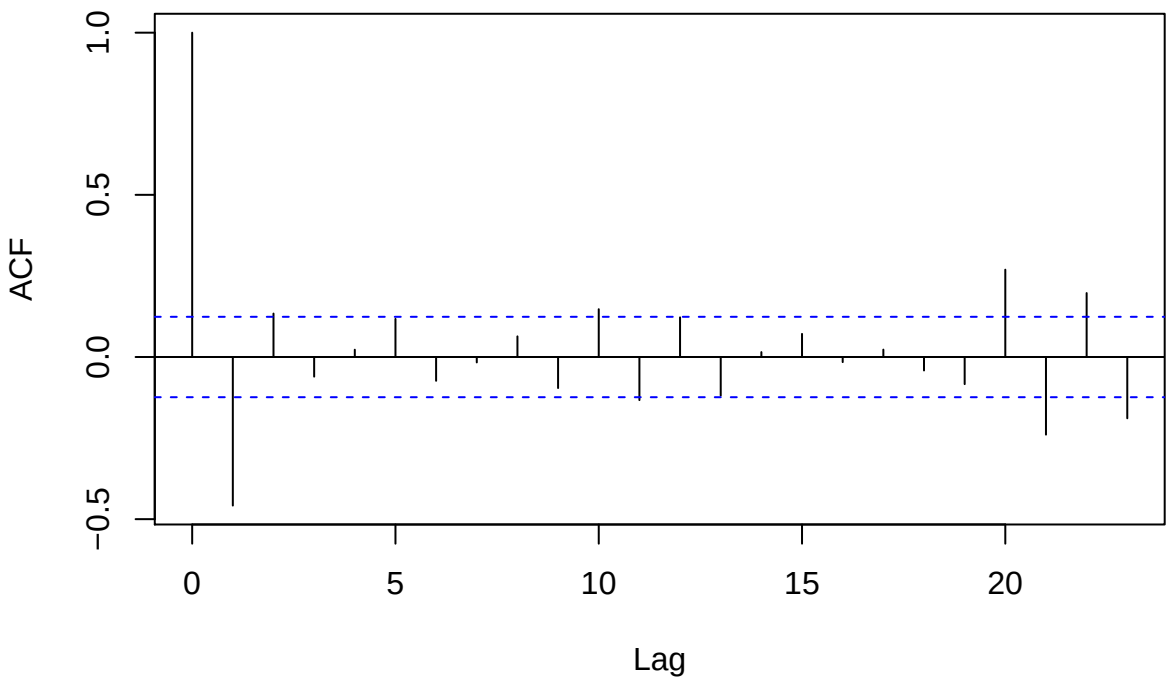
```
plot(test_data$oil, preds, main = "Predictions vs. Actual Prices") #scatterplot of prediction errors
```



This performs moderately well in predictive terms, and we can expect prediction inaccuracy of `0.11`. Predictive power is lost for longer time horizons, though we can see the model at the later time points picks up on the changed volatility.

We inspect the autocorrelation plot.

```
acf(oil_price_d) #plot acf
```



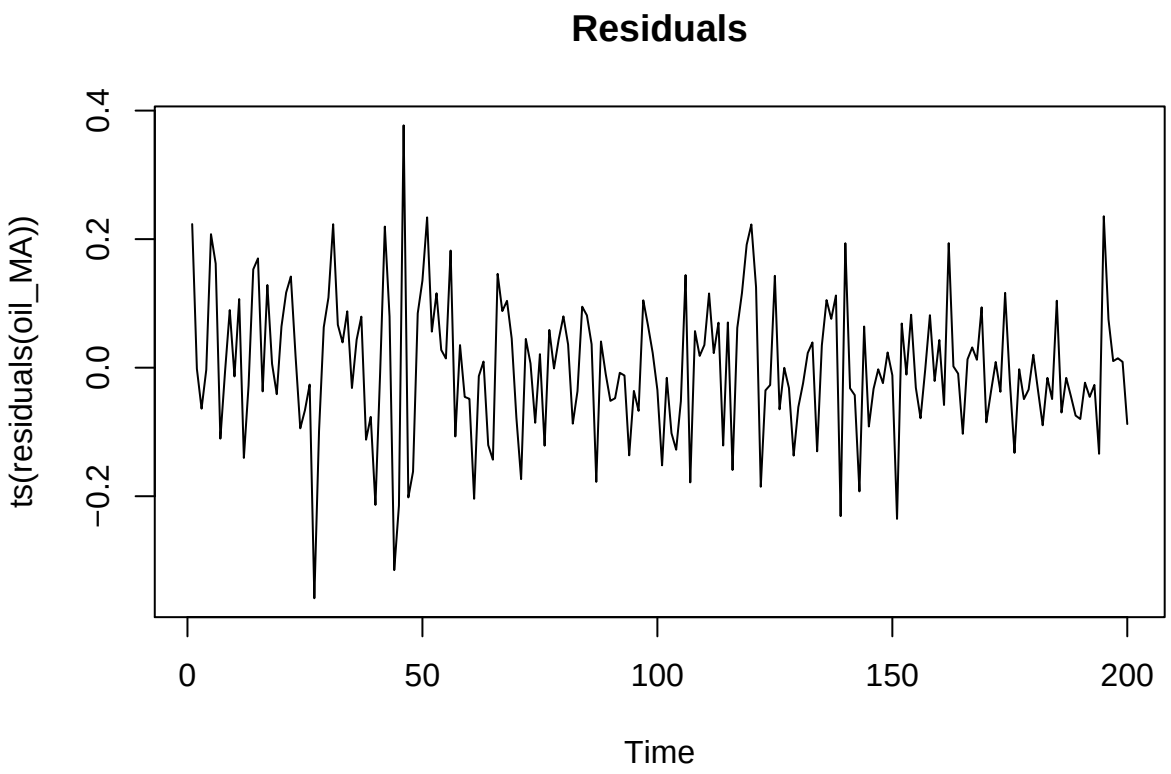
There appears to be a significant autocorrelation with lag 1, so incorporating a temporal dependence component may make the model perform better. We first model a 1-step autoregressive component

$$y_t = \boldsymbol{\beta}^T \mathbf{f}_t + \alpha y_{t-1} + \epsilon_t$$

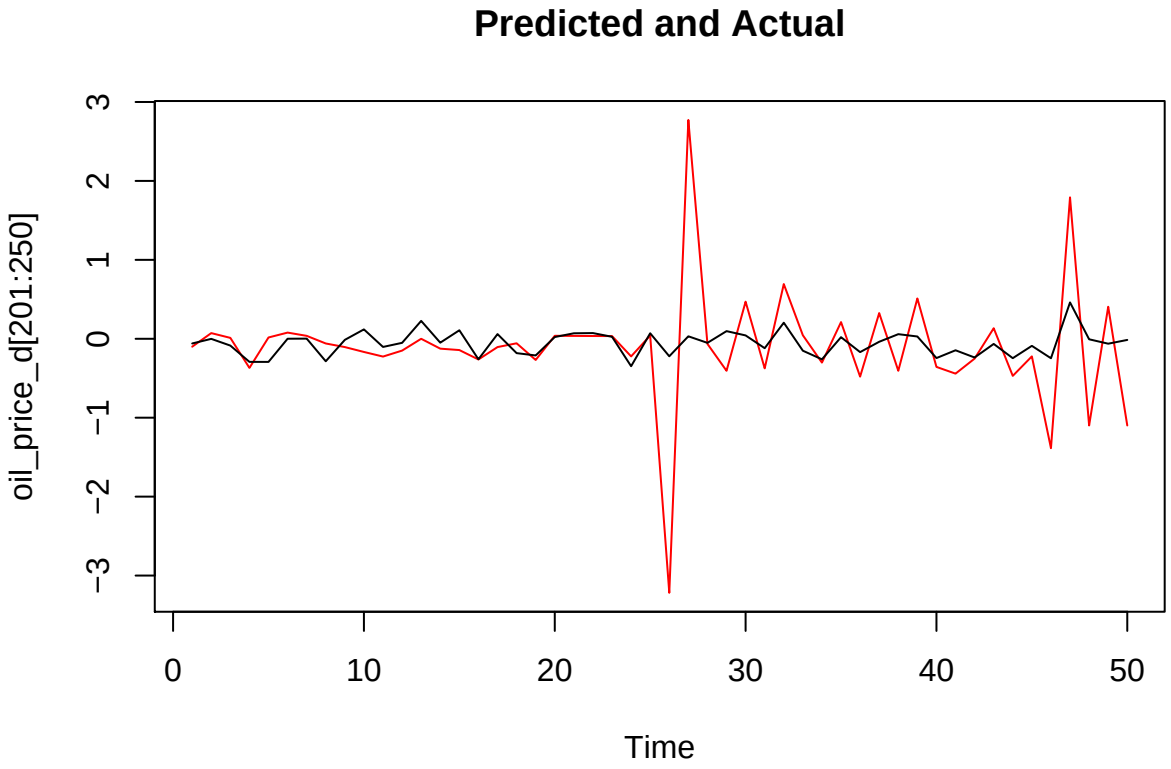
```
oil_price_d_lag <- c(oil_price_d[-1],0) #specify 1-lagged series
model_data <- data.frame(oil = oil_price_d, factor_series, lag = oil_price_d_lag) #group series as data
train_data <- model_data[1:200,] #select first 200 as training
test_data <- model_data[201:250,] #select last 250 as testing

oil_MA <- lm(oil ~ ., data = train_data, na.action = NULL) #fit linear model

mean(residuals(oil_MA)^2) #Mean Squared Error
```

```
## [1] 0.01171281
```

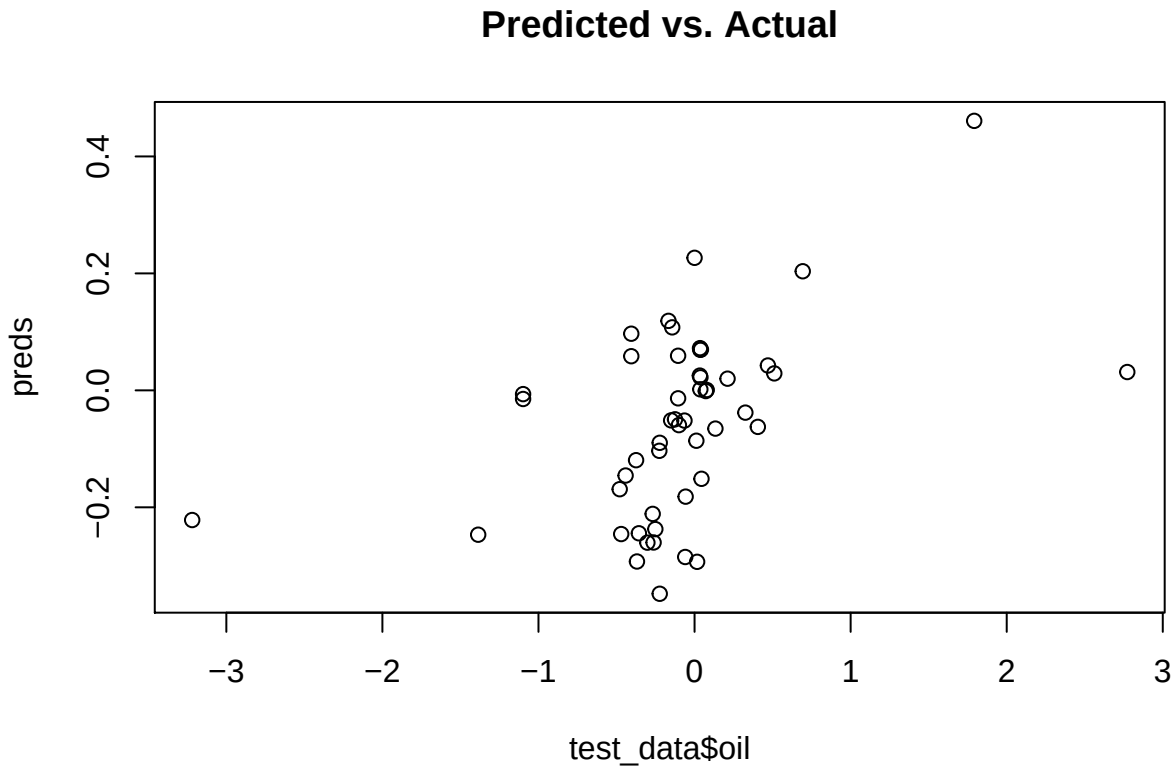
```
plot(ts(residuals(oil_MA)), main = "Residuals") #residual plot
```




```
preds <- predict(oil_MA,  newdata = test_data)
ts.plot(oil_price_d[201:250], col = "red", main = "Predicted and Actual") #plot observed series in red
lines(ts(preds)) #overlay predicted series
```



```r
plot(test_data$oil, preds, main = "Predicted vs. Actual") #scatterplot of prediction errors
```



This does not seem to give us much improvement - here our expected error is similar.

Classifying Movements with Factors

We might find more luck in classifying movements in the log-oil price into increases (+1) and decreases (−1).

We encode the changes as $y \in \{-1, +1\}$ depending on their sign, and use the same covariates as in the last problem, incorporating a 1-step lag term. Here, we implicitly ignore our belief that the data are not IID - this is a strong assumption to place on the data, and may affect both the validity and the performance of the model.

Support Vector Machines

```r
change <- as.numeric(sign(oil_price_d)) #encode change as +1 or -1
lag_change <- as.numeric(sign(oil_price_d_lag)) #encode lag series
model_data <- data.frame(change = change, factor_series, lag_change = lag_change)
#group series as dataframe
train_data <- model_data[1:200,] #select first 200 as training
test_data <- model_data[201:250,] #select last 50 as testing
```

We use the `SVM` function from our package.

We aim to find a classification function $f(\boldsymbol{x}, \boldsymbol{\beta}, \alpha)$ by minimising

$$||(\boldsymbol{\beta}^T, \alpha)||^2 + \sum_t e_t$$

such that

$$\forall t, y_t((\boldsymbol{\beta}^T, \alpha)(\mathbf{f}_t, y_{t-1})^T) + e_t \geq 1, e_t \geq 0$$

where e_t is the distance of the point $(\mathbf{f}_t, y_{t-1})$ into the margin.

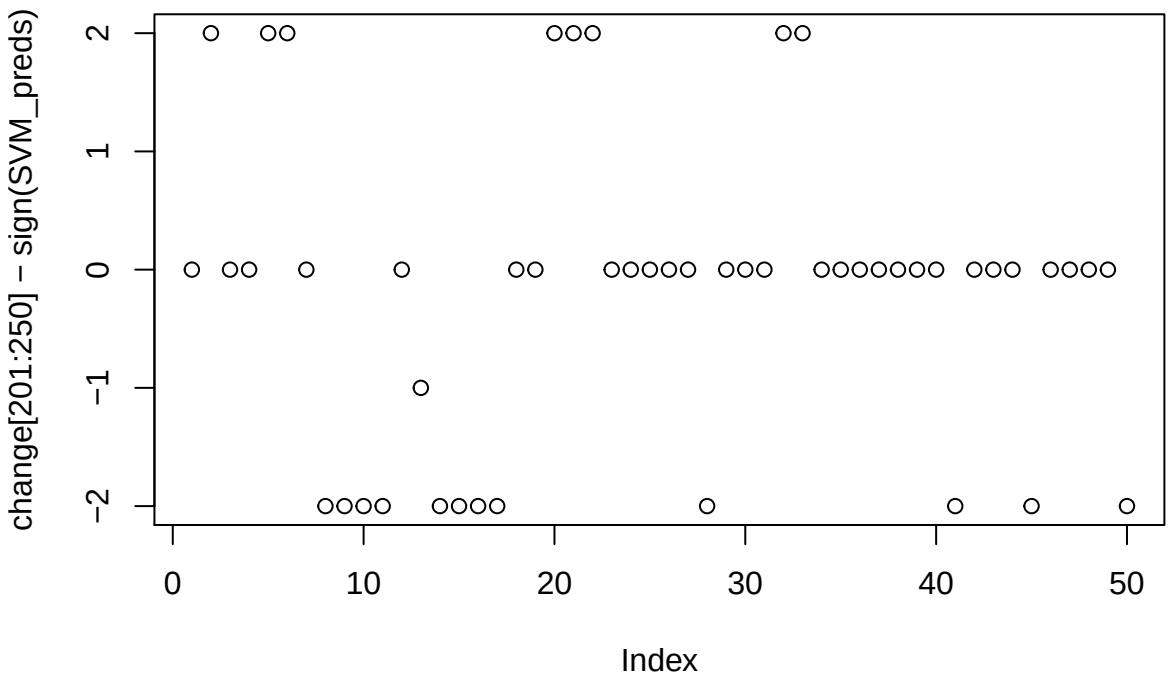
```
w <- SVM(X = train_data[,-1], y = train_data$change,
         max_it = 1e3, eta_0 = 1, alpha = 0.9, c = 0.9) #fit SVM
w #print coefficients
```

```
##  [1] -0.0804302310 -0.0453498289 -0.0045101914 -0.0063259283  0.0015873594
##  [6] -0.0037269749 -0.0004802676 -0.0009214566 -0.0007763616  0.0033364857
## [11] -0.0324392385
```

```
SVM_preds <-  as.matrix(test_data[,-1]) %*% w
head(sign(SVM_preds))
```

```
##     [,1]
## 201   -1
## 202   -1
## 203    1
## 204   -1
## 205   -1
## 206   -1
```

```
plot(change[201:250] - sign(SVM_preds)) #plot class residuals over time
```




```
sum(change[201:250] != sign(SVM_preds))/length(SVM_preds)
```

```
## [1] 0.42
```

```
#calculate percentage of misclassified points
```

This method does not perform particularly well, and is likely no better than classifying with a simple $Y \sim Bernoulli(0.5)$ assignment.

Bibliography

Manel Youssef & Khaled Mokni, 2019. "Do Crude Oil Prices Drive the Relationship between Stock Markets of Oil-Importing and Oil-Exporting Countries?," Economies, MDPI, Open Access Journal, vol. 7(3), pages 1-22, July.

Pattern Recognition and Machine Learning, Christopher Bishop

Principal Components Analysis, Ian T. Jolliffe

S&P 500: Portfolio Optimisation via Factor Analysis

Introduction: Modern Portfolio Theory

Modern Portfolio Theory (firt proposed by economist **Harry Markowitz**) draws on the logical idea that, given the choice between two portfolios with the same expected return, you would prefer the portfolio with the smallest variance (or alternatively, the highest average return for a given level or risk). In this context, the variance of a portfolio is a direct metric for the portfolio's "risk".

Definition: A *Portfolio* of financial assets is defined as a set of weights $\boldsymbol{w} = (w_1, \ldots, w_p)$ such that $\boldsymbol{w}^T \mathbf{1}_p \leq 1$, where $\mathbf{1}_p$ is the p-length vector of 1's. This definition allows for *short-selling* (where an investor hold a negative quantity of a financial asset). In some cases the additional constraint that $w_i \geq 0$, $\forall i = 1 \ldots p$ which prevents short selling.

The idea of MPT boils down to a constrianed optimisation problem, taking the form

$$\max_{\boldsymbol{w}} \boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{\mu}$$

$$\text{s.t. } \boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{\Sigma} \boldsymbol{w} \leq \sigma_0^2, \quad \boldsymbol{w}^T \mathbf{1}_p \leq 1$$

Where $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ are the mean and covariance of the returns respectively. That is we look for a portfolio that delivers the maximum expected return such that the variance of the portfolio does not exceed σ_0^2. The choice of σ_0^2 quantifies an investors aversion to risk.

In this form (i.e. no constraints on short-selling) there is a well known analytical solution to this optimization problem *see Proposition 2.1* (Bai, Liu, and Wong 2009):

$$\text{If } \sigma_0 B \leq \sqrt{A}, \text{ then } R = \sigma_0 \sqrt{A} \text{ and } \boldsymbol{w} = \frac{\sigma_0}{\sqrt{A}} \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}$$

$$\text{If } \sigma_0 B > \sqrt{A}, \text{ then } R = \frac{B}{C} + D(A - \frac{B^2}{C}) \text{ and } \boldsymbol{w} = \frac{1}{C} \boldsymbol{\Sigma}^{-1} \mathbf{1}_p + D(\boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} - \frac{B}{C} \boldsymbol{\Sigma}^{-1} \mathbf{1}_p)$$

Where R denotes the expected return of the portfolio, $A = \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}$, $B = \mathbf{1}_p^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}$, $C = \mathbf{1}_p^T \boldsymbol{\Sigma}^{-1} \mathbf{1}_p$ and $D = \sqrt{\frac{C\sigma_0^2 - 1}{AC - B^2}}$.

Factor Analysis for Portfolio Optimisation

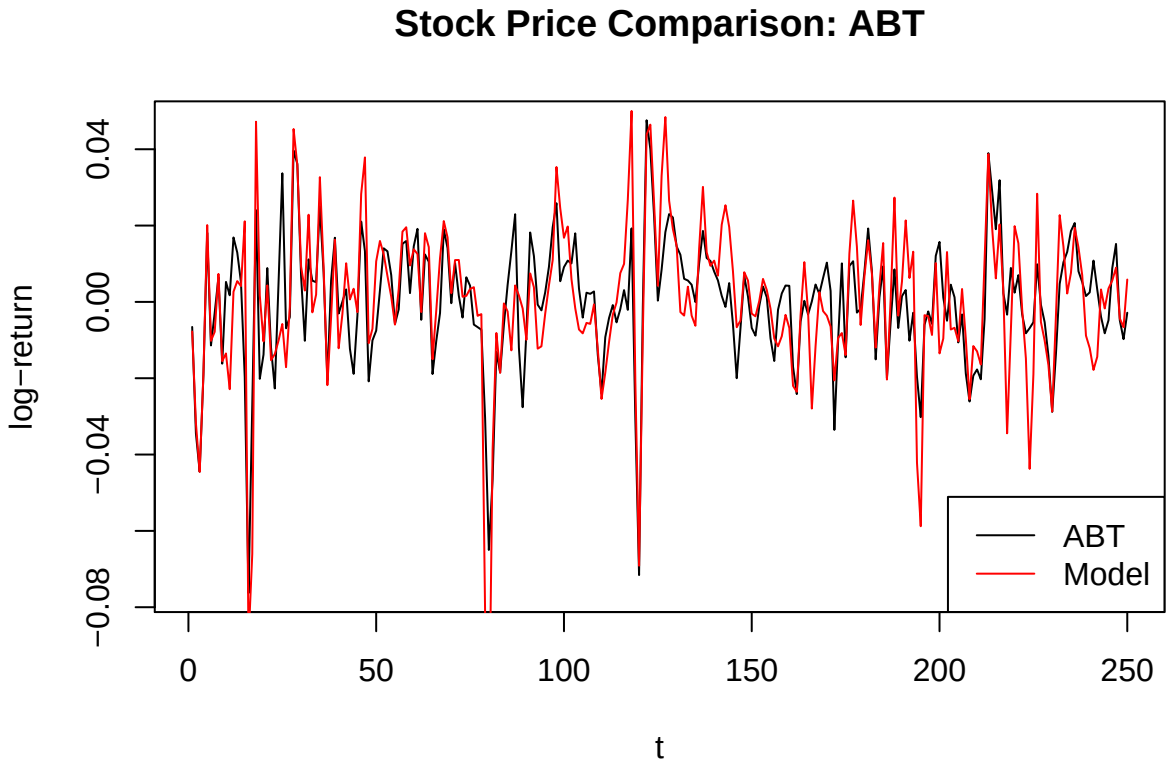
Following a similar approach to that described by (Chen, Huang, and Pan 2015), we use a *factor model* to model our data the S&P 500 stock prices. We have access to daily closing-bid prices from 2014 to 2019.

```
##   X.1       date   ABT  ABBV    ACN    ACT   ADBE  ADT   AES   AFL
## 1   1 2018-12-31 72.33 92.19 141.01 21.400 226.24 6.01 14.46 45.56
## 2   2 2018-12-28 71.09 91.12 139.82 21.359 223.13 6.11 14.27 44.95
## 3   3 2018-12-27 70.63 89.91 140.41 21.250 225.14 6.16 14.29 44.98
## 4   4 2018-12-26 69.62 89.04 139.01 21.560 222.95 6.29 14.28 44.14
## 5   5 2018-12-24 65.56 84.16 133.67 21.030 205.16 6.01 13.82 42.33
## 6   6 2018-12-21 67.27 84.92 137.20 21.420 208.80 6.21 14.45 43.23
```

The data is modelled as

$$y_{it} = \mu_i + \boldsymbol{\lambda}_i^T \boldsymbol{f}_t + u_{it} \Leftrightarrow \boldsymbol{Y} = \boldsymbol{\mu}\boldsymbol{1_n}^T + \boldsymbol{\Lambda}^T \boldsymbol{F} + \boldsymbol{U}$$

With $\boldsymbol{Y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$, $\boldsymbol{F} = (\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n)$, $\boldsymbol{U} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n)$, $\boldsymbol{\Lambda} = (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_p)$. Taking the 2-day log-returns from 2016-01-01 to 2017-01-01 as a training set, we can now fit our model.



After fitting a model with 10 factors, we can see from this plot that the model roughly fits the first timeseries corresponding to ABT. Of course, it is important to remember that to properly assess the performance of the model we would need to compute the **testing error**. Unfortunately, since our model contains latent variables that must be estimated, we cannot evaluate our model at a point in the test set.

As a replacement metric for performance, we will construct a covariance matrix from the model and use MPT to find a portfolio. We will compare this to the simplistic strategy of dividing all capital equally amongst the assets.

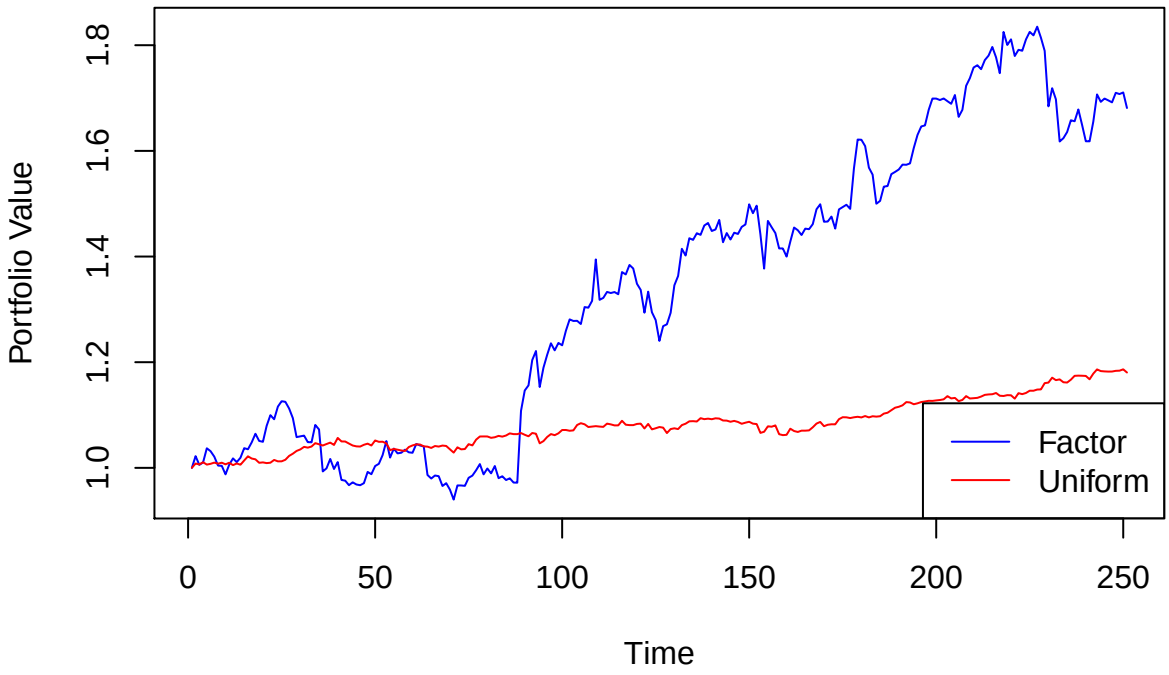
Here will will make use of a **quadratic programming** algorithm found in the `quadprog` package. This makes use of the dual method proposed by (Goldfarb and Idnani 1983) and solves the problem:

$$\min_{\boldsymbol{w}} \frac{1}{2} \boldsymbol{w}^T \boldsymbol{\Sigma} \boldsymbol{w} - \boldsymbol{\mu}^T \boldsymbol{w}$$

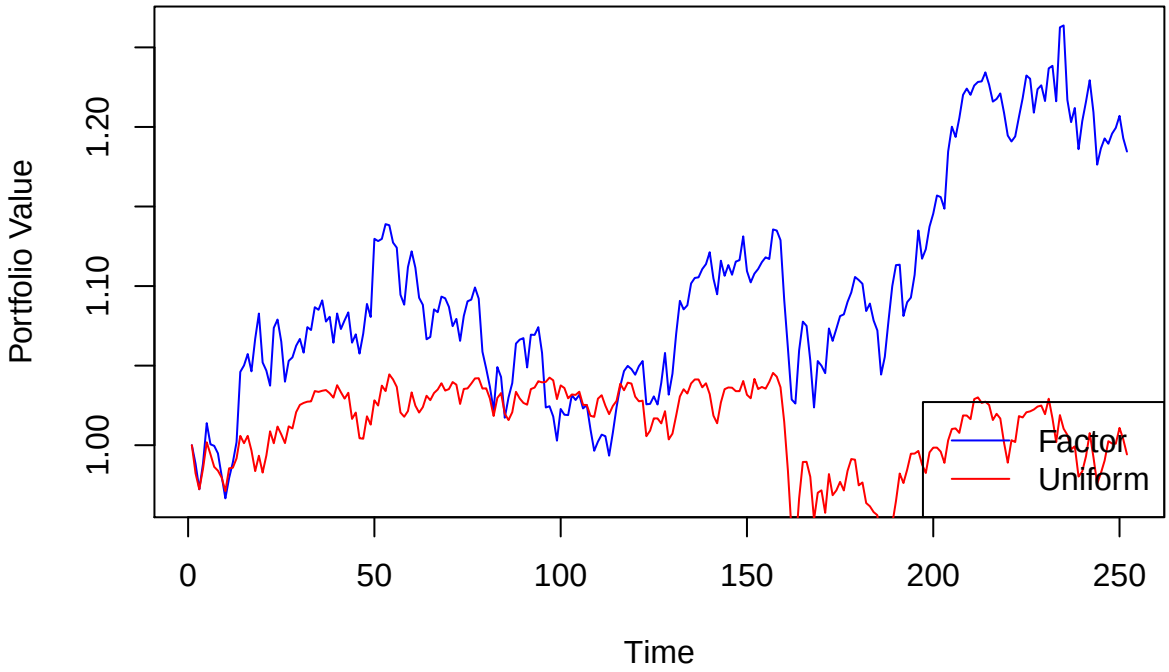
$$\text{S.t. } \boldsymbol{A}^T \boldsymbol{w} \geq \boldsymbol{b}_0$$

Instead of treating the variance as a constraint, this approach aims to minimize/maximize the variance/expected return of the portfolio directly. In the following implementation we allow the variance term to be multiplied by a positive constant $\lambda = 10$. A higher value of λ will result in a solution with a smaller variance.

Train: 2016-01-01 to 2017-01-01 Test: 2017-01-01 to 2018-01-01



Train: 2014-01-01 to 2015-01-01 Test: 2015-01-01 to 2016-01-01



Conclusion: Over seperate time periods the portfolios constructed via factor analysis seem to out-perform the uniformly distributed portfolio. This suggests there is a chance that our model contains some predictive power. **However**, without a proper metric such as the testing error, we do not have sufficient evidence to claim our model is successful at representing the stock market data.

References

Bai, Zhidong, Huixia Liu, and Wing-Keung Wong. 2009. "ENHANCEMENT of the Applicability of Markowitz'S Portfolio Optimization by Utilizing Random Matrix Theory." *Mathematical Finance* 19 (4): 639–67. https://doi.org/10.1111/j.1467-9965.2009.00383.x.

Chen, Binbin, Shih-Feng Huang, and Guangming Pan. 2015. "High Dimensional Mean–Variance Optimization Through Factor Analysis." *Journal of Multivariate Analysis* 133: 140–59. https://doi.org/https://doi.org/10.1016/j.jmva.2014.09.006.

Goldfarb, D., and A. Idnani. 1983. "A Numerically Stable Dual Method for Solving Strictly Convex Quadratic Programs." *Mathematical Programming* 27. https://doi.org/https://doi.org/10.1007/BF02591962.

Gaussian Processes for Time Series

Bayesian Time Series Analysis

Given some time series data $D = \{(x_i, y_i)\}$ where x_i is say a time point and y_i is some reading at this time, we consider it as a regression problem:

$$y_i(\boldsymbol{x}) = f_i(\boldsymbol{x}) + \eta,$$

where f is unknown and and η is a random additive noise process, with the goal in mind to evaluate a form of f and infer a probability distribution for y given an input x. Throughout we will use the notation $\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})]$.

Gaussian Processes

To conducdct this analysis we consider a Gaussian Process (a good intro to which lies in (Rasmussen 2003)), which is completely defined by the mean function and covariance function:

$$m(\boldsymbol{x}) = \mathbb{E}(\boldsymbol{f}(\boldsymbol{x}))$$
$$k(\boldsymbol{x}, \boldsymbol{x'}) = \mathbb{E}[\{\boldsymbol{f}(\boldsymbol{x}) - m(\boldsymbol{x})\}\{\boldsymbol{f}(\boldsymbol{x'}) - m(\boldsymbol{x'})\}]$$

and we write the process as: $f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x'}))$, and for simplicity often set the mean function to zero.

Assumptions

A Gaussian process is defined as a collection of random variables where $\boldsymbol{x}$ is the input of time series indices, and in our case $\boldsymbol{f}(\boldsymbol{x})$ represents a time series. Hence we are assuming that we fullfil a consistency requriement, that is:

$$(y_1, y_2) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \implies y_1 \sim \mathcal{N}(\boldsymbol{\mu_1}, \boldsymbol{\Sigma}_1),$$

for the relevant submatrix $\boldsymbol{\Sigma_1}$. That is, the examination of a large set of variables does not effect the distribution of each of its subsets, a criterion automatically fulfilled when we specify a kernel function for the covariance matrix.

The model

A Gaussian process can be obtained from a Bayesian liner regression model $f(\boldsymbol{x}) = \phi(\boldsymbol{x})^\top \boldsymbol{w}$ where $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma_p})$ - then the means and kernel functions become:

$$m(\boldsymbol{x}) = \mathbb{E}[\boldsymbol{f}(\boldsymbol{x})] = \phi(\boldsymbol{x})^\top \mathbb{E}[\boldsymbol{w}] = 0$$

and

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \mathbb{E}[\boldsymbol{f}(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x'})] = \phi(\boldsymbol{x})^\top \Sigma_p \phi(\boldsymbol{x'})$$

For our time series data we will use the most widely used kernel function used for this class of problem (Roberts et al. 2013), the squared exponential function:

$$k(x_i, x_j) = \sigma^2 \exp\left[-\frac{1}{2l}(x_i - x_j)^2 \right]$$

With hyperparameters σ and l, choices of which can result in very different curves, σ effectively controls the gain of the function, and l can be thought of as a smoothing parameter. It is known, (Rasmussen 2003), that this corresponds to a regression with infinite basis functions.

Fitting

Prediction using Noisy Observations

It makes sense, for time series data, to assume input time indeces $x_i = i, i = 1, \ldots$ are noiseless, and that the randomness comes with the values at each timestamp y_i. Let X be the input set for our known values, and X_* be those of the values we wish to predict with a function $\hat{\boldsymbol{f}}$ .So given a dataset we have access to noisy values $y = f(\boldsymbol{x}) + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$, hence the prior induced is:

$$cov(\boldsymbol{y}) = K(X, X) + \sigma_n^2 \boldsymbol{I},$$

and we can now, using consistency, write down the joint distribution of observed values $\boldsymbol{y}$ and function values $\boldsymbol{f}_*$:

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}_* \end{bmatrix} = \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} K(X, X) + \sigma_n^2 \boldsymbol{I} & K(X, X_*) \\ K(X_*, X) & K(X_* X_*) \end{bmatrix} \right)$$

Using results on conditional Gaussian distributions (Bishop 2006), we can write down predictive equations:

$$\boldsymbol{f}_* \mid X, \boldsymbol{y}, X_* \sim \mathcal{N}\left(\hat{\boldsymbol{f}}_*, cov(\boldsymbol{f}_*) \right),$$

where

$$\hat{\boldsymbol{f}}_* = K(X_*, X)[K(X, X) + \sigma_n^2 \boldsymbol{I}]^{-1} \boldsymbol{y},$$
$$cov(\boldsymbol{f}_*) = K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 \boldsymbol{I}]^{-1} K(X, X_*).$$

The next section will be dedicated to exploring this scheme using $\hat{\boldsymbol{f}}_*$, the conditional mean, as our predictive function on some real data - the `Fit_GP` function in our package gives a method of how to fit a Gaussian Process to time series data.

Real Data

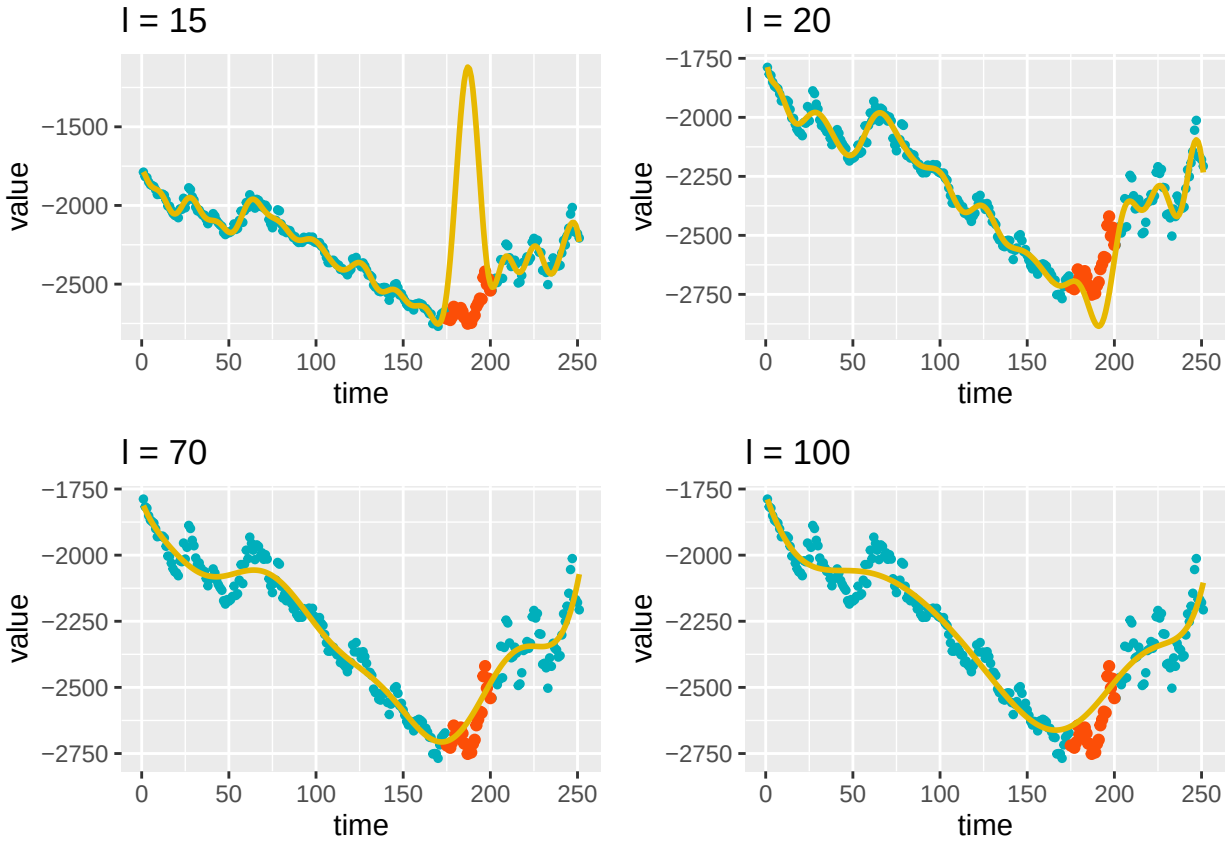
Casting our minds back to our factor analysis, we now present fits of a Gaussian process to a reduction of the undifferenced SP500 multivariate time series to a single variate factor of 251 values through time, fitted using the `myPCA` function in the `portopt` package. Fitting a Gaussian Process to the factor series corresponding to the greatest varience of the original data will allow us to analyse underlying behaviour of the SP500 data as a whole. We will now see how Gaussian processes perform when trying to interpolate and predict missing data.

Choice of Kernel parameter l for Predicting Missing Data

The problem set up here is very simple, suppose we are in the setting that a section of our factor series data has gone missing, either through collection, computer, or human error and we wish to try and learn the behaviour of our time series in order to interpolate to fill the gap in the readings. Recall our kernel function:

$$k(x_i, x_j) = \sigma^2 \exp\left[-\frac{1}{2l}(x_i - x_j)^2\right]$$

To simplify the problem let's fix our the gain parameter to $\sigma = 200$, and let's learn l to perform an interpolation for a missing set of 25 consecutive readings. Let's look at how our model prediction function, fit using the `Fit_GP` function in the `portop` package , looks for various choices of l. Here the Gaussian Process has been fit on the factor series readings with a section of 25 readings missing from $175, \ldots, 200$ for which our conditional mean serves as our prediction function, we plot it against the values the process is fitted on in blues, and the missing values in red:



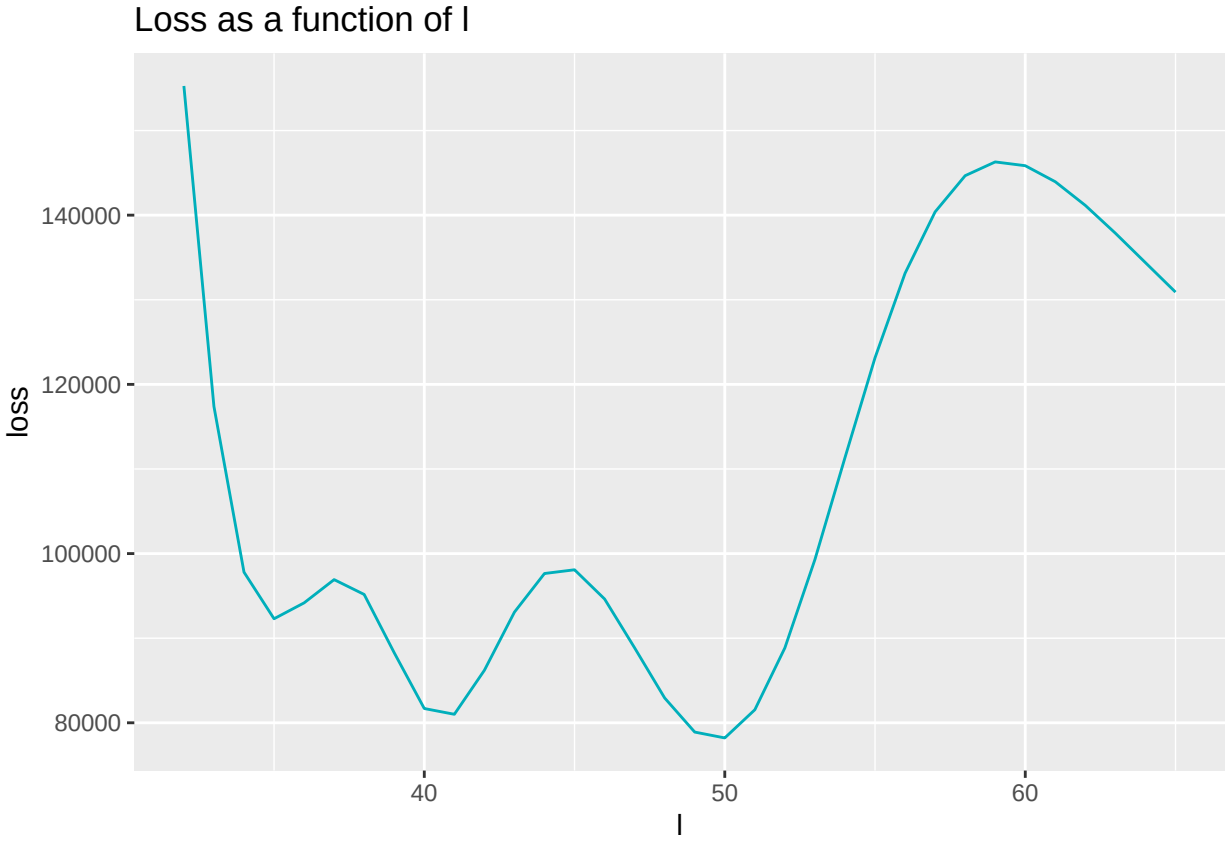
We see that increasing l decreases the flexibility of the prediction function. An interpretation of l given in (Rasmussen 2003) is that it can be thought of as a 'decay time' relative to σ, that is, how quickly should expect the process to change between readings - this interpretation is backed up by these observations as for lower values of l we see the prediction function moving further from the known values.

Let's define a loss function for our choice of l, it is defined only over the missing data indexed $j = 1, \ldots, 25$, we take the squared loss of the predictions:

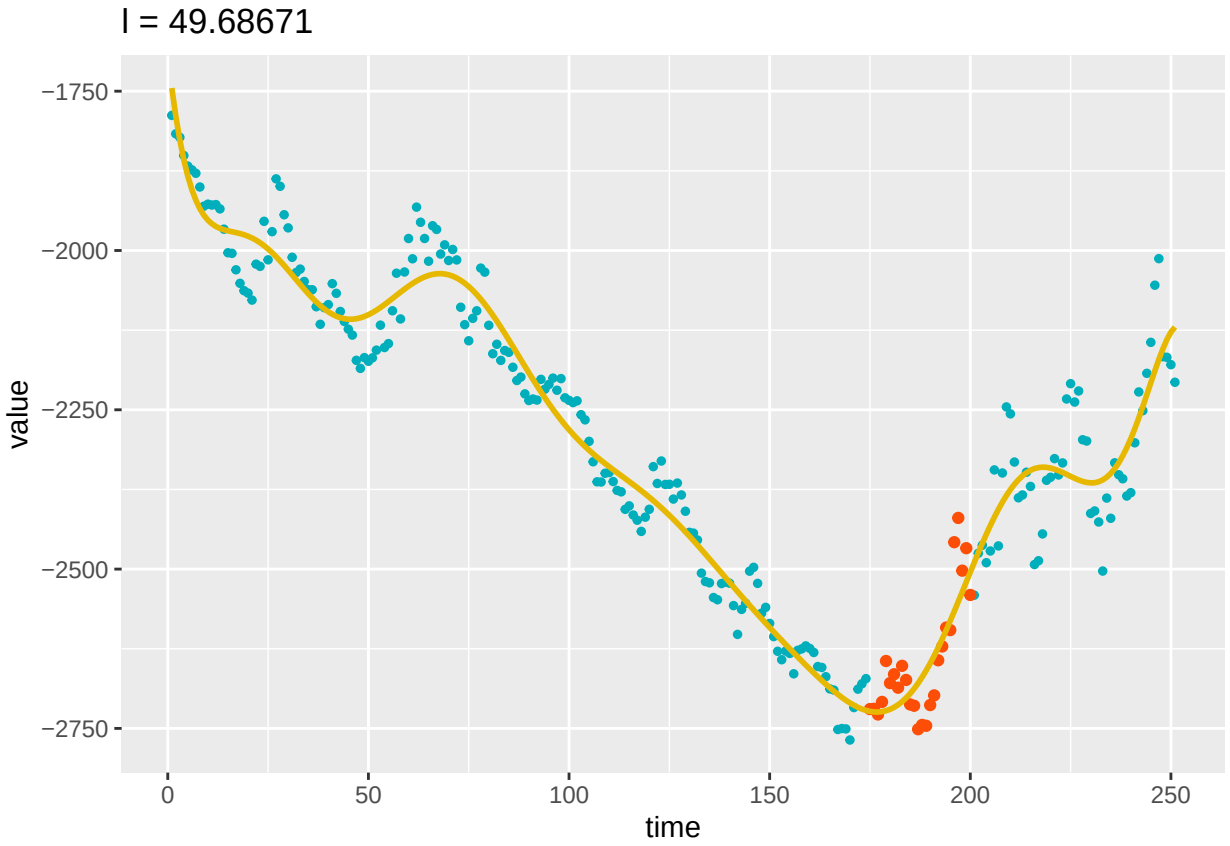
$$L(l) = \sum_{j=1}^{25} (y_{true}^{(j)} - f_j(\boldsymbol{x}))^2,$$

where $f_i(\boldsymbol{x})$ is our predictive mean at timestamp i. Let's plot this over reasonable values of l - preliminary analysis shows a loss which grows for extreme values of l so plotting over $l \in [30, 70]$ will include the relevant optimal points:

```
## [1] 2437969344
```



We see that our loss has a multiple local optima, a phenomenon frequently observed with optimisation of kernel paramenters (Rasmussen 2003), it appears we have a global minimum. Running an optimisation with the 'Brent' method indeed tells us that our squared loss is minimised for $l = 49.68671$ attaining a value of 2437969344:



Testing

So we have learned a value of l over the training set $i = 175, \ldots, 200$, let's see if our learned kernel generalises to fit unseen gaps:

Here we see our value for l tested on various sets of missing data with loss values 1778024179, 2917764245, 194541423, and 6415421161, respectively - Suggesting that the method of learning l does not generalise very well.

Conclusions

We have seen how taking a Bayesian view of modelling a time series allows us to model it as a Gaussian Process equivalent to using infinite basis functions in a regression setting, modelling as such leads to a rich arsenal for inference, notably a predictive conditional mean, variance and likelihood obtained using results from the rules of conditional Gaussian distributions. Whilst the attempt at a method to model and predict missing data values has proved to generalise weakly, an insight was gained as to how the choice of l for the squared exponential kernel effects the fit of the Gaussian Process. Further methods to explore for such problems could include utilising the predictive variance and conditional likelihood into the optimisation of of the parameters of the kernel function such as those suggested in (Roberts et al. 2013).

References

Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning*. springer.

Rasmussen, Carl Edward. 2003. "Gaussian Processes in Machine Learning." In *Summer School on Machine Learning*, 63–71. Springer.

Roberts, Stephen, Michael Osborne, Mark Ebden, Steven Reece, Neale Gibson, and Suzanne Aigrain. 2013. "Gaussian Processes for Time-Series Modelling." *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 371 (1984). The Royal Society Publishing: 20110550.