Financial Data Project: Forecasting Oil Movements with Factors

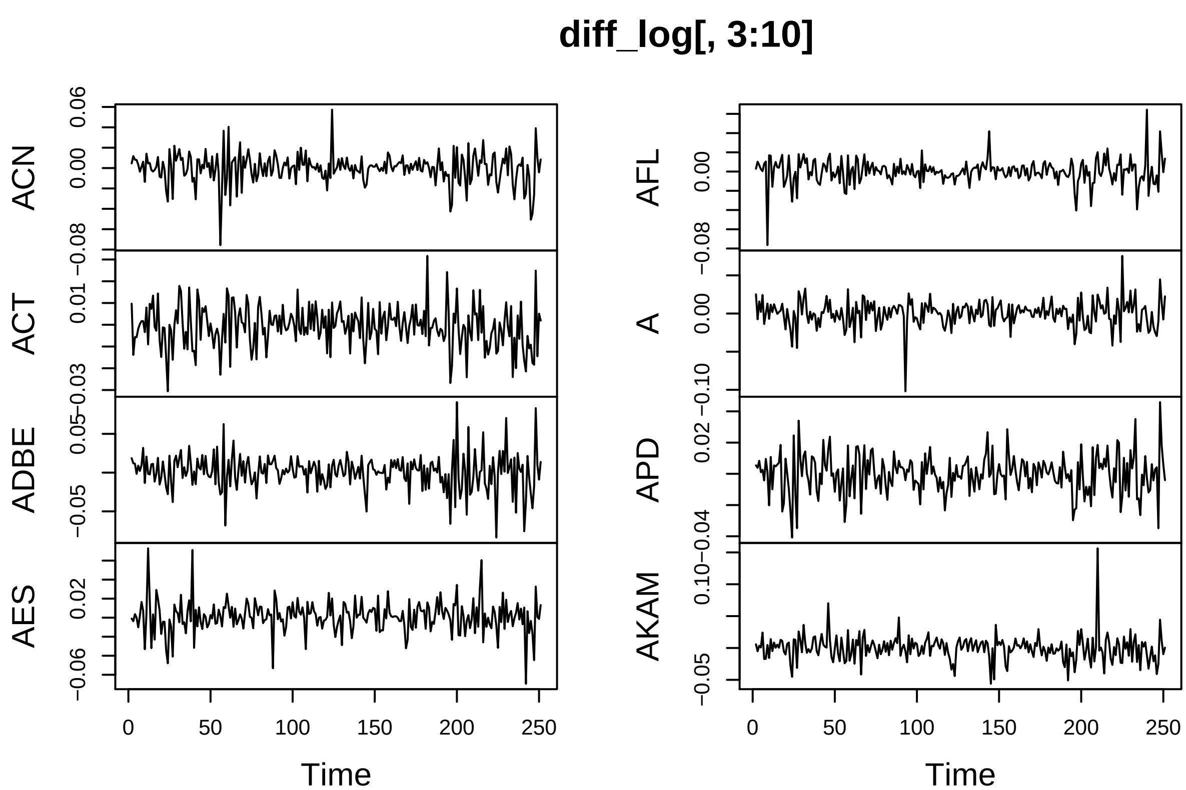
Dom Owens

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.

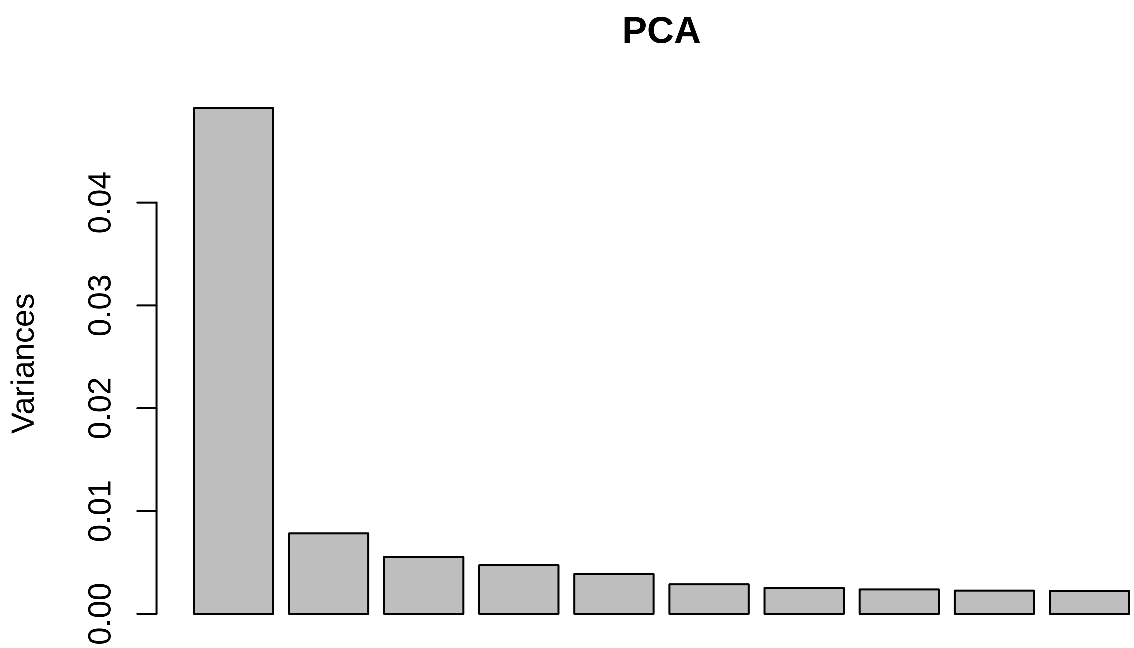
```
close_data_2018 <- subset(oil_close_data, select = -c(X.1, DATE, DCOILWTICO, ADT))
#drop date and oil, ADT
log_close <- log(close_data_2018) #take log
diff_log <- diff(ts(log_close)) #diff series
plot(diff_log[, 3:10]) #plot first 8 series
```



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
```

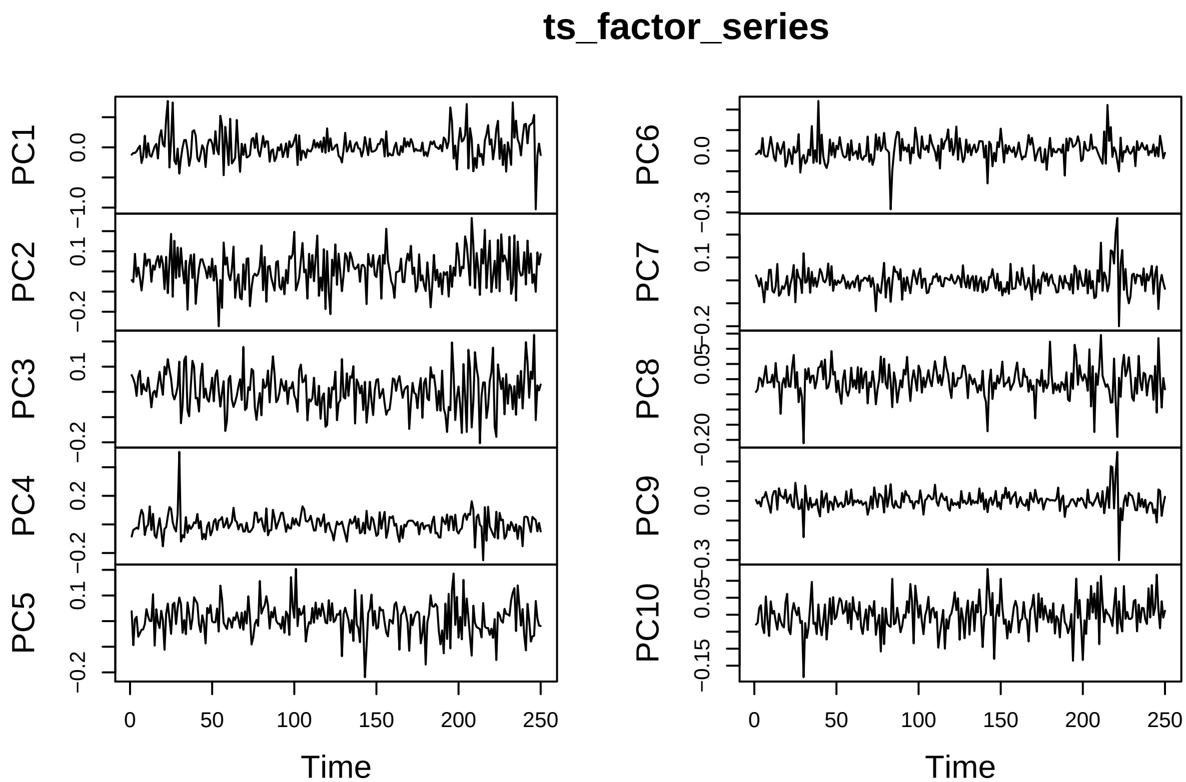



```r
head(PCA$sdev, 30) #view standard deviations of first 30 components
```

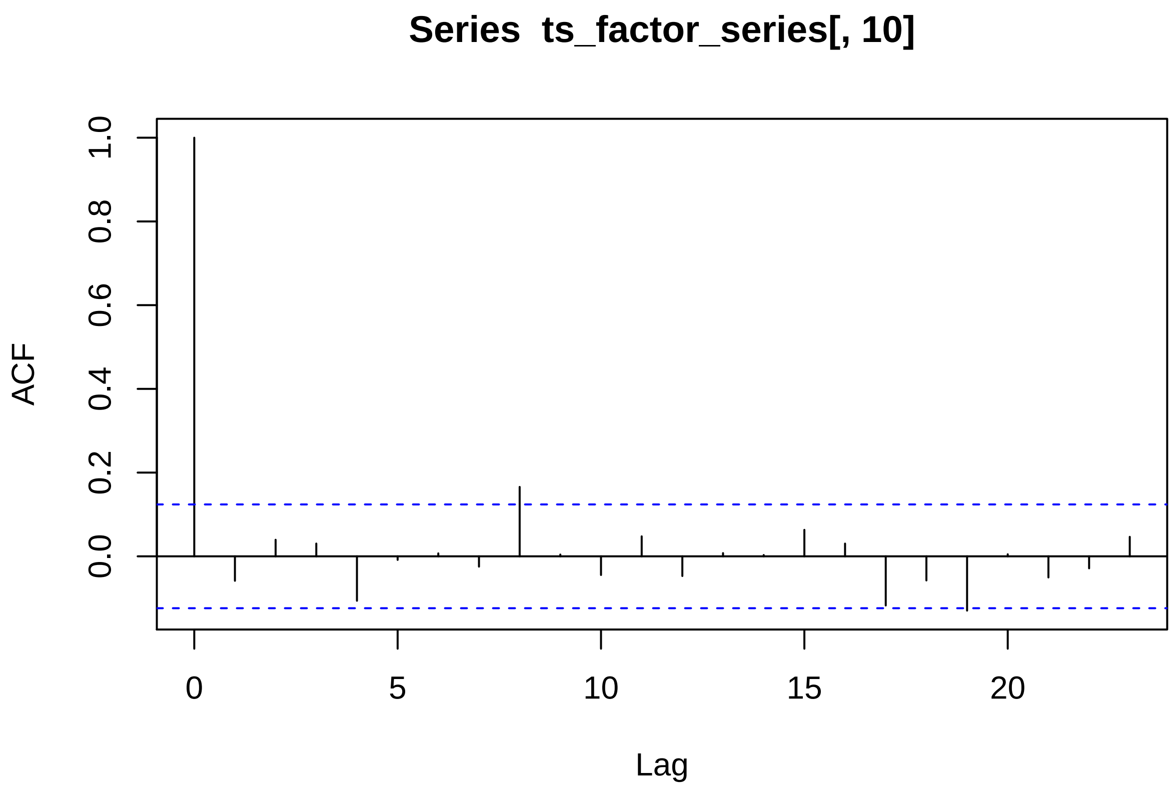
```
##  [1] 0.22176908 0.08847104 0.07451144 0.06877110 0.06226909 0.05361267
##  [7] 0.05034662 0.04874833 0.04756765 0.04706843 0.04440906 0.04257630
## [13] 0.04116579 0.04015350 0.03917405 0.03825732 0.03715776 0.03645910
## [19] 0.03583580 0.03514537 0.03488699 0.03442400 0.03424505 0.03375637
##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

We arbitrarily select the first 10 components.

```r
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
```



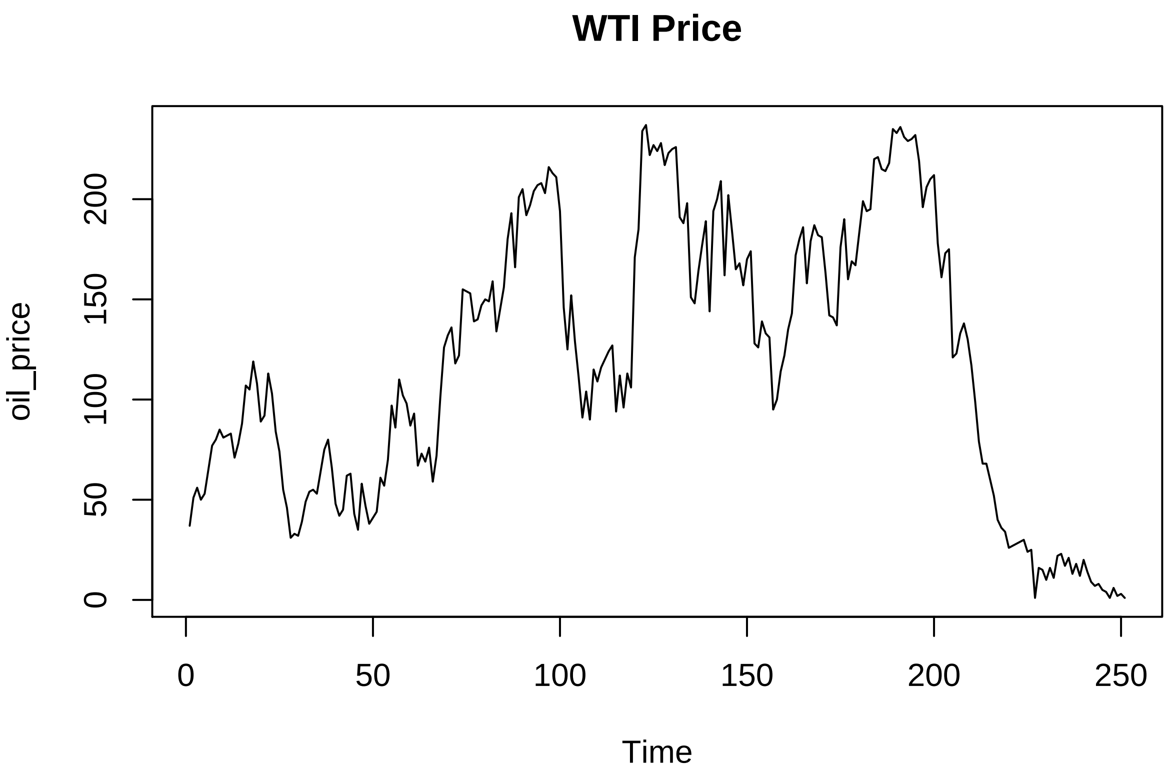

```
acf(ts_factor_series[,10]) #plot autocorellations
```



Forecasting oil prices using factors

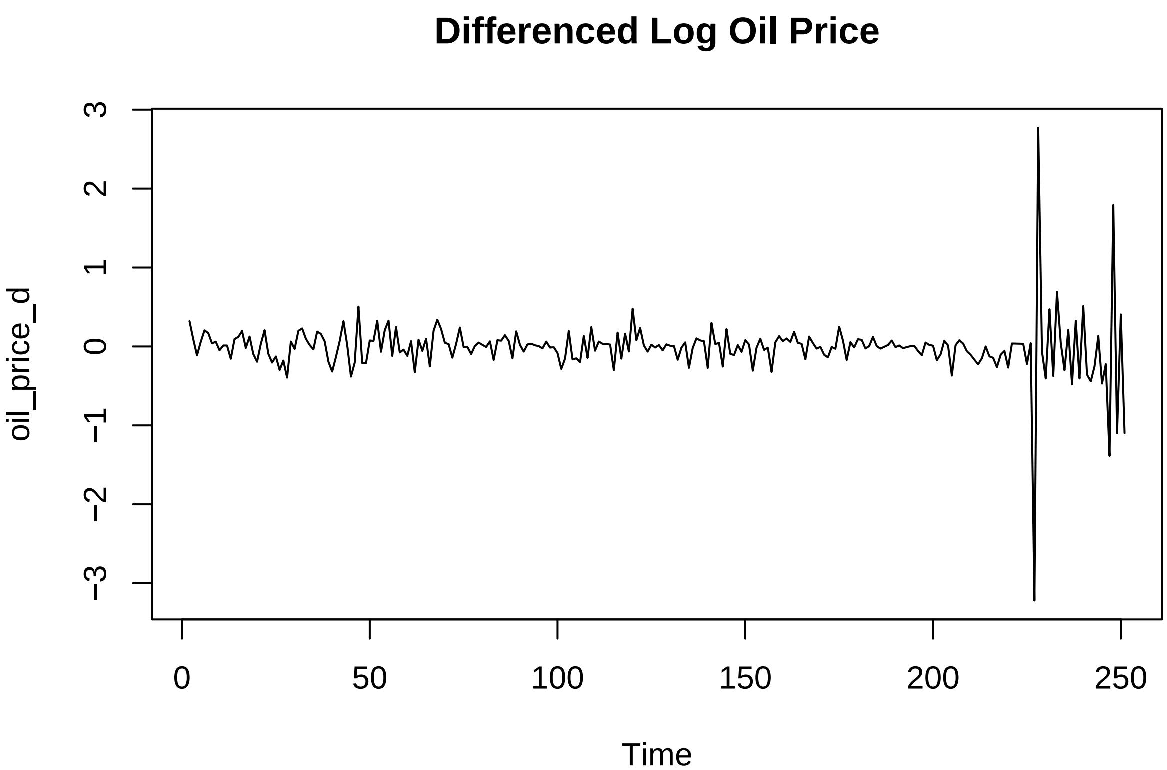
Suppose we wish to forecast the spot price of a different, but related, asset. We might think there is some relationship between the price of US-produced crude oil WTI and the S&P 500; we construct a regression on our factor representation to describe this.

```
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 $\mathbf{f}_t$:

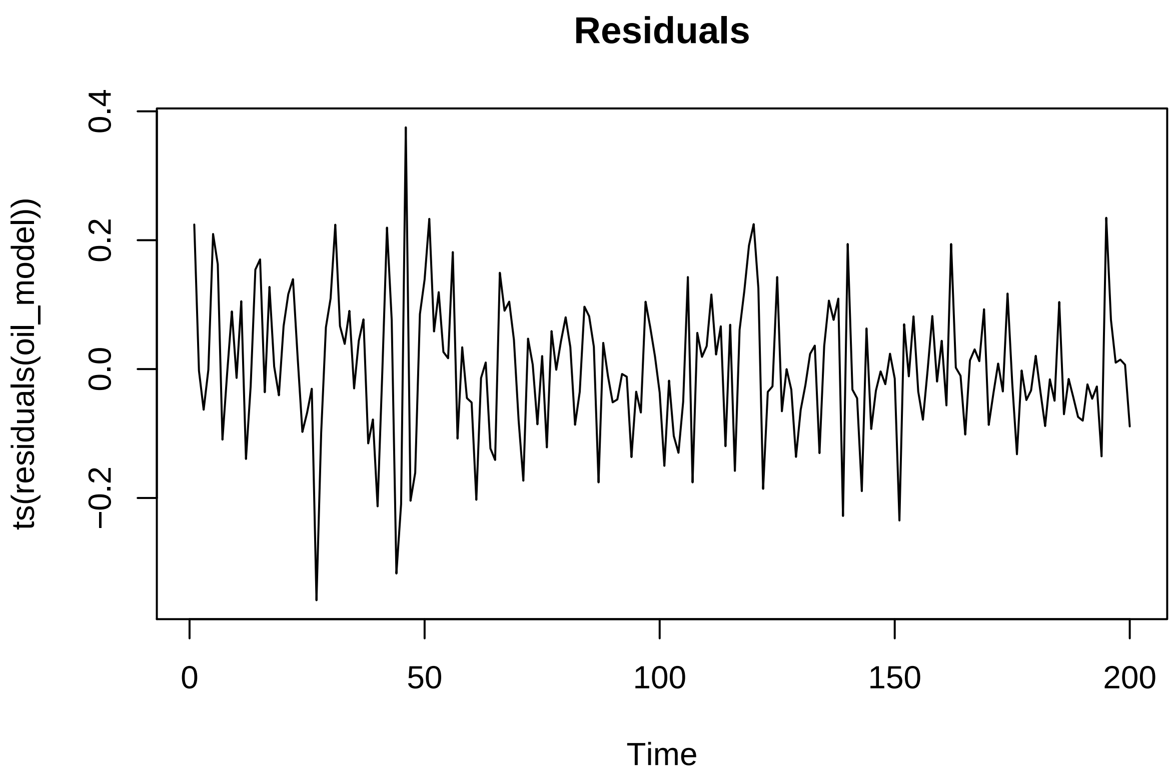
$$y_t = \boldsymbol{\beta}^T \mathbf{f}_t + \epsilon_t$$

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

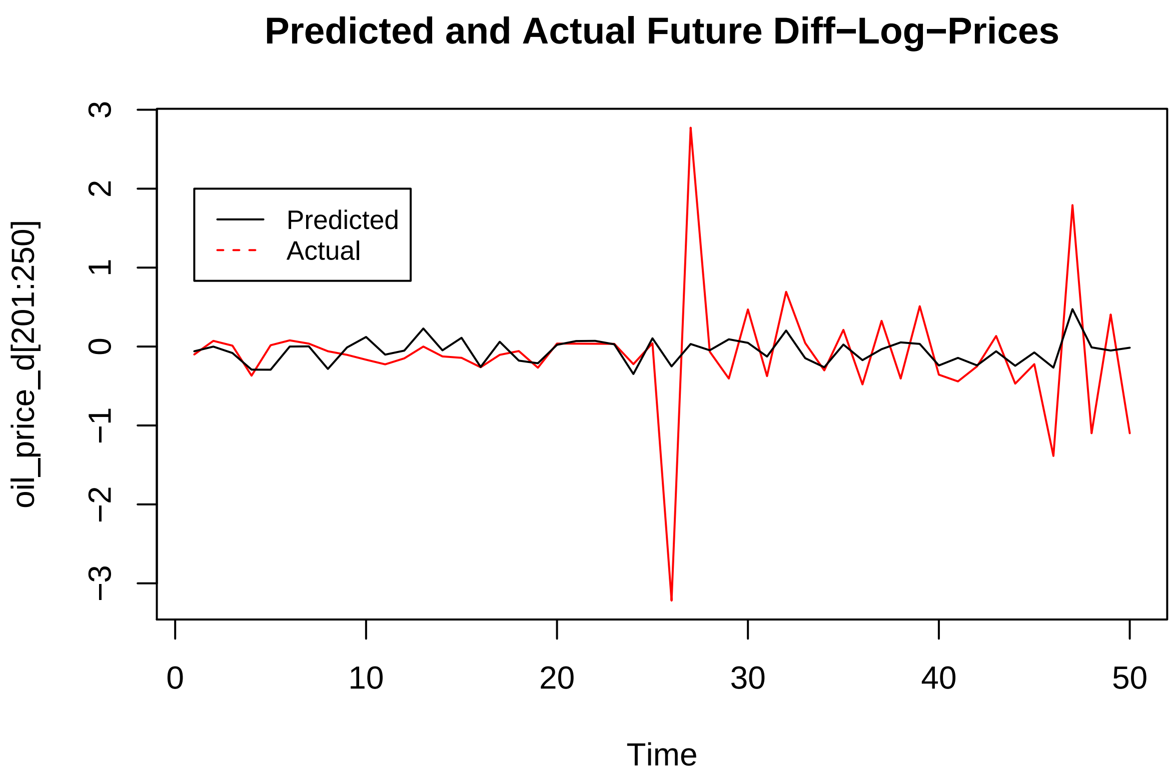
```
model_data <- data.frame(oil = oil_price_d, factor_series )#group series as dataframe
train_data <- model_data[1:200,] #select first 200 as training
test_data <- model_data[201:250,] #select last 250 as testing

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

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

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

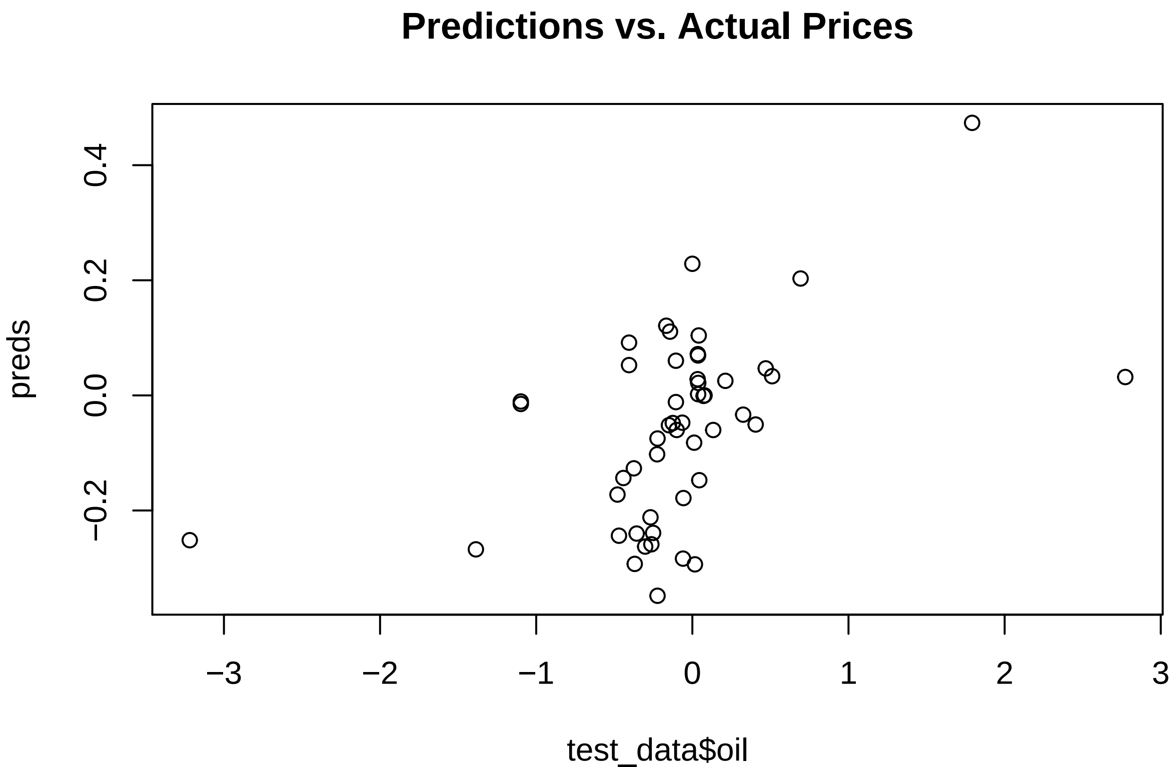
```
plot(ts(residuals(oil_model)), main = "Residuals") #residual plot
```




```r
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
```



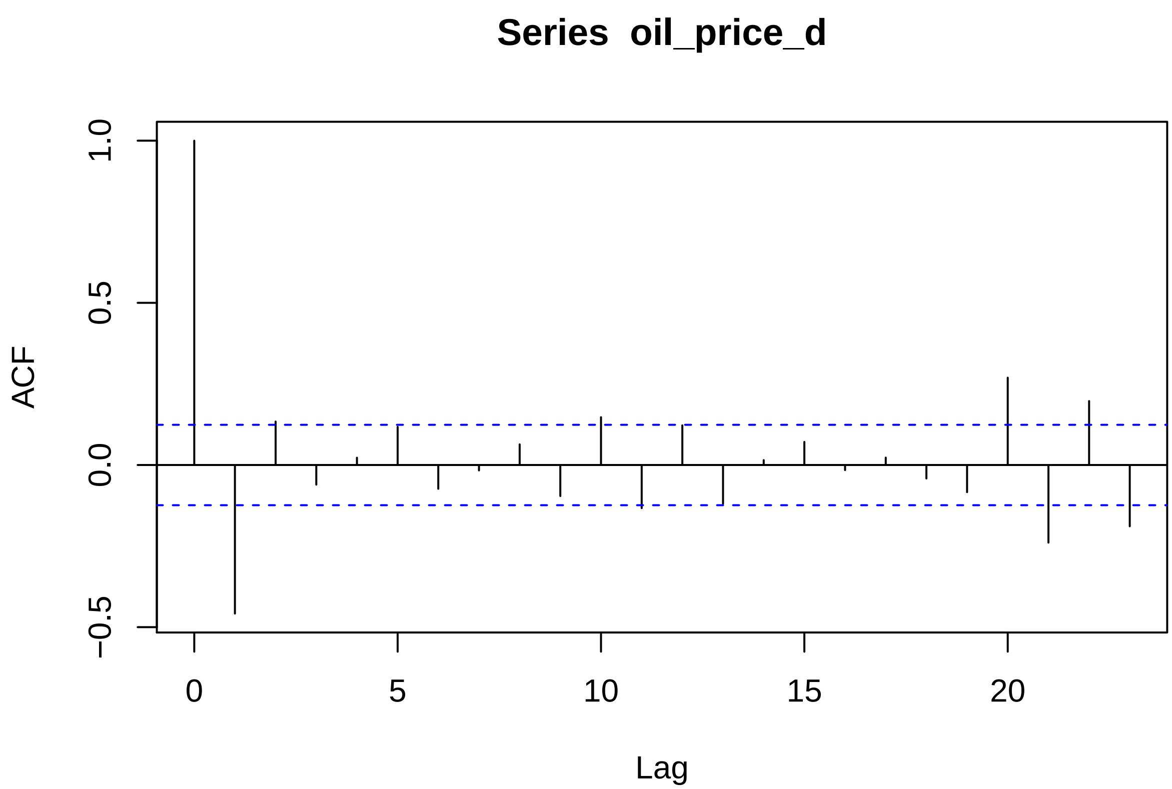
```r
plot(test_data$oil, preds, main = "Predictions vs. Actual Prices") #scatterplot of prediction errors
```



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.

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



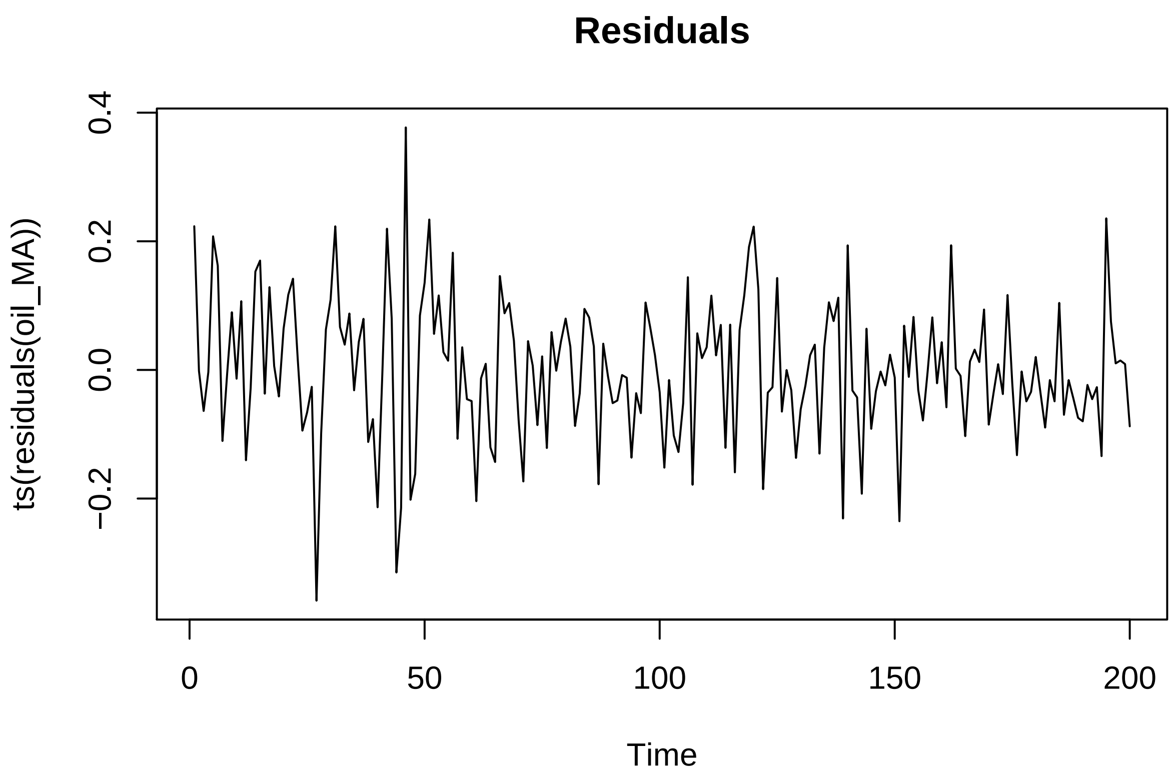
There appears to be a significant autocorrelation with lag 1, so incorporating a temporal dependence component may make the model perform better. We first model a 1-step autoregressive component

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

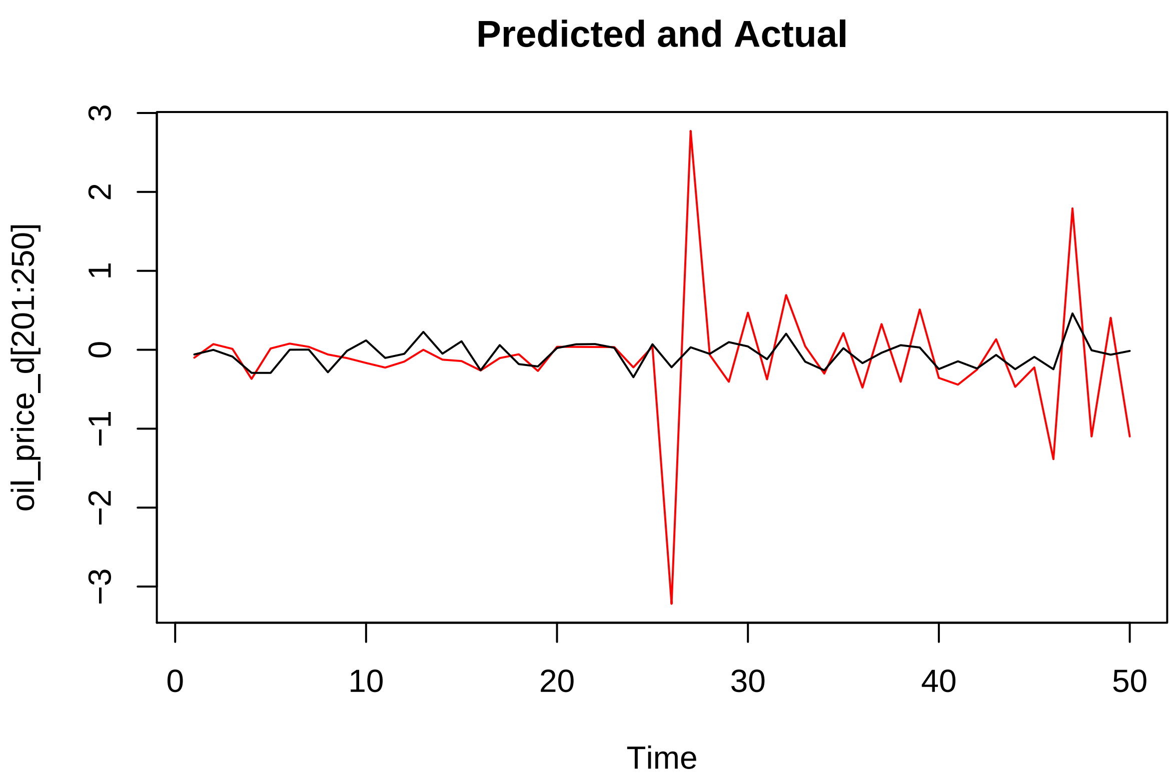
```
oil_price_d_lag <- c(oil_price_d[-1],0) #specify 1-lagged series
model_data <- data.frame(oil = oil_price_d, factor_series, lag = oil_price_d_lag) #group series as data
train_data <- model_data[1:200,] #select first 200 as training
test_data <- model_data[201:250,] #select last 250 as testing

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

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

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

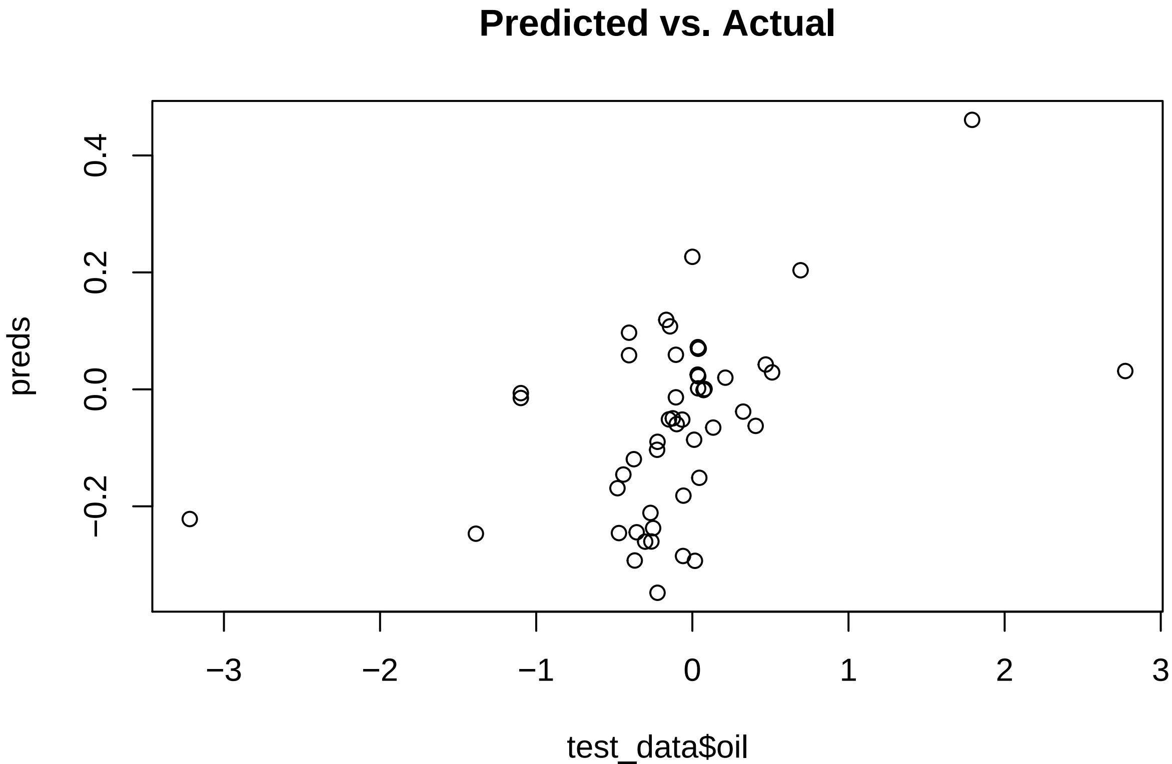
```
plot(ts(residuals(oil_MA)), main = "Residuals") #residual plot
```




```r
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.

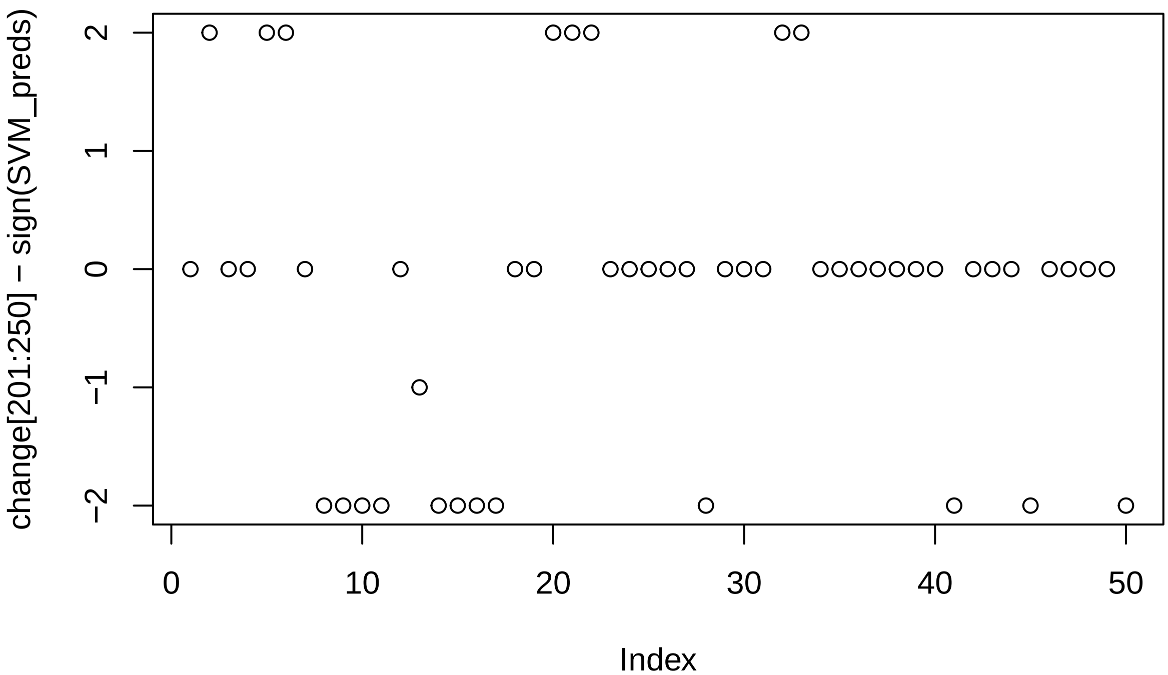
```
w <- SVM(X = train_data[,-1], y = train_data$change,
         max_it = 1e3, eta_0 = 1, alpha = 0.9, c = 0.9) #fit SVM
w #print coefficients
```

```
##  [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