Final Project

Objective

Our objective is to build an algorithm that can predict the movement of 1 month WTI Oil future contracts by using the spot price as a predictor. We will be using particle filters, a form of Sequential Monte Carlo, to build our models and predictions.

Terminology

A future contract grants the holder the right to X units of some commodity for $\$Y$ in Z months. If held until expiry, the commodity will be delivered physically. Future contracts give individuals the ability to hedge their commodity position by locking in a future price, or speculate on the directional movement of the commodity.

The spot price of the commodity refers to the price a commodity can be bought and delivered at in the present.

Dataset

We took out data from Investing.com. We used historical data for Spot prices and 1 month future contract prices April 15th, 2010 to April 15th, 2020 to create our data set.

Initially, we noticed that the data did not align perfectly, with there being missing days between our spot price data and future price data. We removed all data points that were not shared between the two data sets.

https://www.investing.com/commodities/crude-oil-historical-data https://www.investing.com/currencies/wti-usd-commentary

Particle Filtering

Particle filtering is a sequential Monte-Carlo (MC) method that seeks to predict a hidden state variable ($\mathbf{x}$) from a series of observations ($\mathbf{y}$). $p(\mathbf{x}_0)$ is the initial state of the distribution, the transition equation is $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, and $p(\mathbf{y}_t|\mathbf{x}_t)$ is the marginal distribution of the observation. Using Bayes' theorem, we derive an expression for $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$, the marginal distribution of the hidden state variable from the observations:

$$p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1})}{\int p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1})d\mathbf{x}_t}$$

$$p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$$

We can also compute $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ recursively via the marginal distribution:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})d\mathbf{x}_{t-1}$$

To find the expected value of $E[f(x_t)]$:

$$E[f(\mathbf{x}_t)] = \int f(\mathbf{x}_{0:t}) p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{0:t}$$

Do we need intermediate steps here?

$$E[f(\mathbf{x}_t)] = \frac{\int f(\mathbf{x}_{0:t}) p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{0:t}}{\int p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{0:t}}$$

To evalute this integral, we introduce $w(x_{0:t})$, the importance weight. The importance weight is equal to:

$$w(x_{0:t}) = \frac{p(x_{0:t}|y_{1:t})}{\pi(x_{0:t}|y_{1:t})}$$

the importance sampling factor. The weight is very important in a particle filter algorithm as it allows us to pick which states are more likely than others and reduce down potential states before resampling. This weight, and subsequently, the importance sampling factor relies on (in our project with crude oil prices) the probability of a tomorrow's future price, given today's spot price divided by π, which is denoted by a factor that assesses different variables that influence the movement of tomorrow's future price.

$$E[f(\mathbf{x}_t)] = \frac{\int f(x_{0:t}) w(x_{0:t}) \pi(x_{0:t}|y_{1:t}) dx_{0:t}}{\int w(x_{0:t}) \pi(x_{0:t}|y_{1:t}) dx_{0:t}}$$

Because we are operating under a MC framework, we can create an approximation for this integral:

$$E[f(\mathbf{x}_t)] \approx \frac{\sum_{i=1}^{i=N} f(x_{0:t}^{(i)}) w(x_{0:t}^{(i)})}{\sum_{j=1}^{j=N} w(x_{0:t}^{(j)})} = \sum_{i=1}^{i=N} f(x_{0:t}^{(i)}) w_t^{*(i)}$$

wIs there a reason the indices differ between the sums?

$$p(x_{0:t}|y_{1:t}) \propto p(y_t|x_{0:t}, y_{1:t-1}) p(x_{0:t}|y_{1:t-1})$$

$$= p(y_t|x_t) p(x_t|x_{0:t-1}, y_{1:t-1}) p(x_{0:t-1}|y_{1:t-1})$$
$$= p(y_t|x_t) p(x_t|x_{t-1}) p(x_{0:t-1}|y_{1:t-1})$$

Recalling $w(x_{0:t}) = \frac{p(x_{0:t}|y_{1:t})}{\pi(x_{0:t}|y_{1:t})}$, we can plug in our value for $p(x_{0:t}|y_{1:t})$:

$$w_t^{*(i)} = \frac{p(y_t|x_t^{(i)}) p(x_t^{(i)}|x_{t-1}^{(i)}) p(x_{0:t-1}^{(i)}|y_{1:t-1})}{\pi(x_{0:t}^{(i)}|y_{1:t})}$$

However, we want the weights to update recursively. Defining $\pi(\cdot)$ recursively will help us redefine $w_t^{*(i)}$:

$$\pi(x_{0:t}|y_{1:t}) = \pi(x_t|x_{0:t-1}, y_{1:t}) \pi(x_{0:t-1}|y_{1:t-1})$$

$$w_t^{*(i)} = \frac{p(y_t|x_t^{(i)}) p(x_t^{(i)}|x_{t-1}^{(i)})}{\pi(x_t|x_{0:t-1}, y_{1:t})} \frac{p(x_{0:t-1}^{(i)}|y_{1:t-1})}{\pi(x_{0:t-1}|y_{1:t-1})}$$

$$w_t^{*(i)} = \frac{p(y_t|x_t^{(i)}) p(x_t^{(i)}|x_{t-1}^{(i)})}{\pi(x_t|x_{0:t-1}, y_{1:t})} w_{t-1}^{*(i)}$$

Naive Alogrithm

We began our model by implementing a naive particle filter. Instead of predicting the price of the future contract, we predicted the volatility of the future contract. We did this because volatility is non-linear, for example, a 1 dollar price swing at 10 dollars is significantly larger than a 1 dollar price swing at 100 dollars.

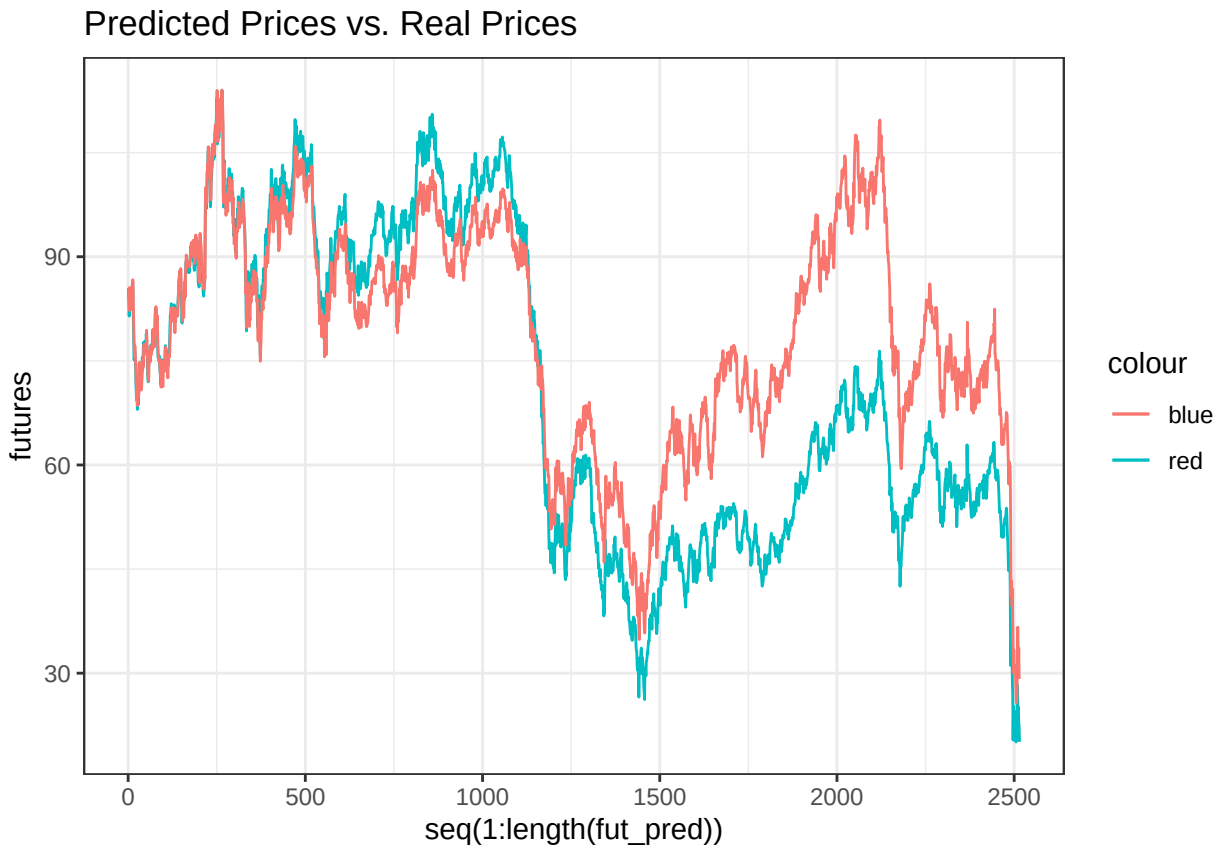
```
set.seed(000)

library(readxl)
Oil_Data <- read_excel("oil_complete.xls")
oil <- na.omit(Oil_Data)

T <- nrow(oil)-1
x_true <- rep(NA, T)
obs <- rep(NA, T)

a <- oil$`Spot Vol`[1:T]
b <- oil$`Future Vol`[1:T]

sx <- sqrt(var(b))
sy <- sqrt(var(a))

for (t in seq(1, T)) {
  x_true[t] <- b[t]
  obs[t] <- a[t]
}


T <- length(obs)
N <- 5000
```

In the naive model, we assume a normal prior distribution.

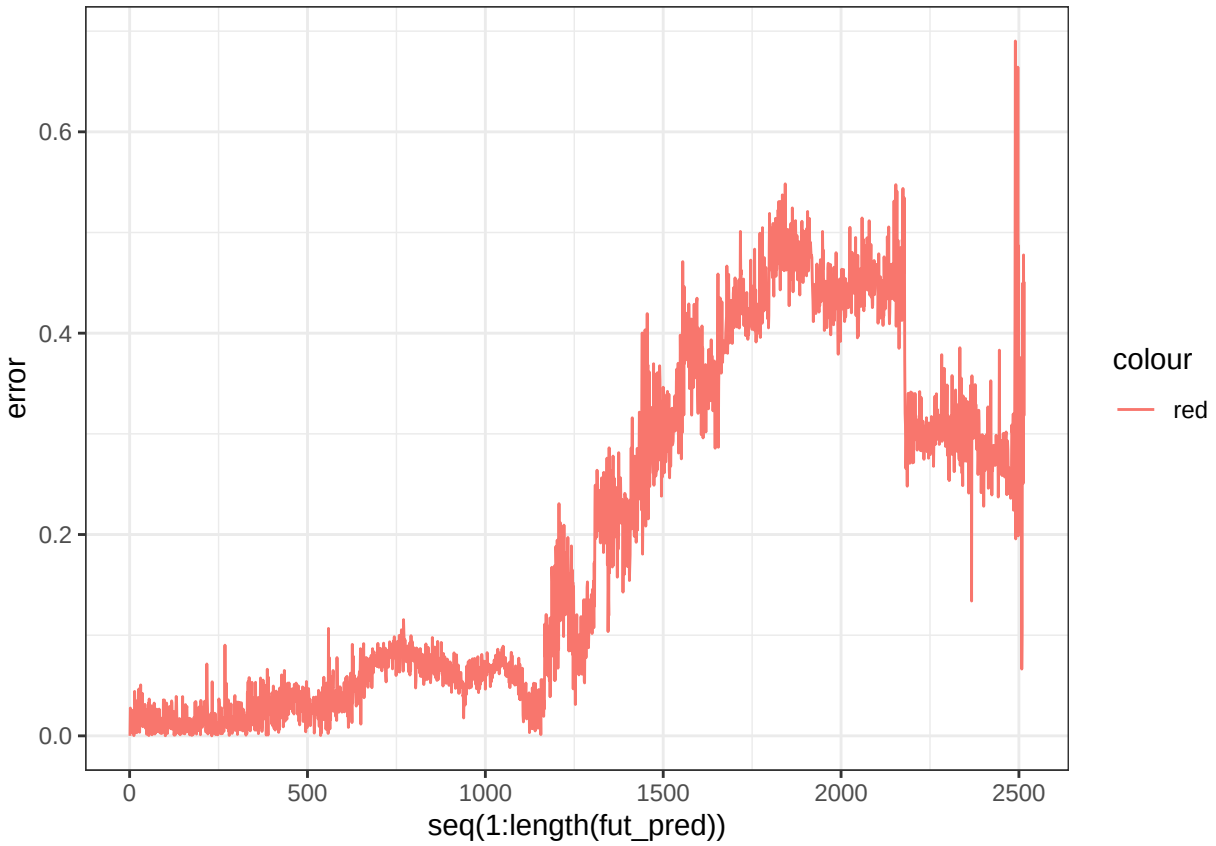
```
x <- matrix(nrow =  N, ncol = T)
weights <- matrix(nrow =  N, ncol = T)

x[, 1] <- rt(N, 300) #Assume a normal prior

weights[, 1] <- dt((obs[1] - x[, 1])/sx, 300) # initial weight set
weights[, 1] <- weights[, 1] / sum(weights[, 1]) # initial weight normalized

x[, 1] <-
  sample(x[, 1],
         replace = TRUE,
         size = N,
         prob = weights[, 1])

for (t in seq(2, T)) {

  x[, t] <- rt(N - x[, t - 1], 300) #sample x from previous x

  weights[, t] <- dt((obs[t] - x[, t])/sx, 300) #get weights from obsveration
  weights[, t] <- weights[, t] / sum(weights[, t]) #normalize weights

  x[, t] <- #resample x using the weights
```

```
    sample(x[, t],
           replace = TRUE,
           size = N,
           prob = weights[, t])
}

fut_pred <- oil$`Future Price`[1]
x_mean <- apply(x, 2, mean) #mean of predictions found

for (i in 1:ncol(x)) { #volatility converted to prices

  new_pred <- fut_pred[length(fut_pred)] *(1+x_mean[i])
  fut_pred <- c(fut_pred, new_pred)

}

futures <- oil$`Future Price`[1:length(fut_pred)]

fut_naive <- NULL

fut_naive <- cbind(fut_naive, Real = futures)
fut_naive <- cbind(fut_naive, Predicted = fut_pred)
```

```
# Real prices vs. Predictions

p1 <- ggplot() + geom_line(aes(y = futures, x = seq(1:length(fut_pred)), colour = "red")) + geom_line(a

error <- abs(futures - fut_pred) / futures

# Absolute error as percentage of price

p2 <-
  ggplot() + geom_line(aes(
    y = error,
    x = seq(1:length(fut_pred)),
    colour = "red"
  ))


p1 + ggtitle("Predicted Prices vs. Real Prices")
```


Predicted Prices vs. Real Prices
futures
90
60
30
0
500
1000
1500
2000
2500
seq(1:length(fut_pred))
colour
blue
red

```
p2
```



Our naive model, as seen in the graph above, is mediocre at predicting oil futures prices, and the predictions seem to be getting worse as time passes. This makes sense, considering that there's a big oil sell-off in the back end of our data. We think there are a couple reasons why this model might not be very successful: (1) our prior is normally drawn into the matrix X, and when looking at the distribution of the data, it does not seem to be normally distributed; and (2) the weights assigned to predict the future price is based off of volatility, which can boost the direction of prediction when looking at larger shifts in the market. To fix these, we will first draw our prior from a log-normal distribution and assign weightages in a different way.

Trading Algorithm

To test our particle filter algorithm, we built a simple trading agorithm to see how much money we would make in the markets. We start with some initial amount of cash. Each day, we make our prediction for the next day. If our predicted value if greater than the current price of the future contact, we buy a contract at today's price. The next day, we sell if the price matches our prediction. If there is an 8% drop, we decide to buy more.

```
prof_ret <- function(range) {
init = oil[1,2]
profit = 0
sellpt = 0
current = init
hold = TRUE
for (i in 1:nrow(oil)){
  if (current < fut_pred[i+1] && hold && abs(x_mean[i+1]) <= range){
    profit = profit + (oil[i+1,2] - current)
    sellpt = oil[i+1, 2]
```

```
    hold = FALSE
  }
  #after repeated trials, 8% drop seems to be the best point to rebuy
  if (!hold && oil[i, 2] <= 0.92*sellpt){
    current = oil[i, 2]
    hold = TRUE
  }
}
tot = profit*1000 #minimum order of 1000 barrels to trade futures
investment = init*1000
roi = tot/investment
roi
tot

return(roi)
}
```

We want to sample through potential ranges to stop trading. This ensures that the model primarily focuses on trading when volatility is small and makes safer trades.

top = 0 range = 0 for (i in seq(from = 0.02, to = 0.20, by = 0.005)){ if (prof_ret(i) > top){ top = prof_ret(i) range = i } } range # This code above took forever to run each time, but it yielded 0.035 as the desired volatility to stop trading

#Shorting Algorithm

```
profit <- NULL
hold = FALSE
for (i in 1:nrow(oil)){
  if (x_mean[i+1] < 0 && i <= 2414){
    shortval = oil[i,2]
    hold = FALSE
  }
  if (!hold && x_mean[i+1] > 0){
    profit = rbind(profit, (shortval - oil[i,2]))
    hold = TRUE
  }
}
tot = sum(profit)
tot*1000
```

```
## [1] 610700
```

```
count = 0
for (i in 1:635){
  if (profit[i,] < 0){
    count = count +1
  }
}
count
```

```
## [1] 61
```

```
((tot*1000 + 85510)/85510)^(1/10)-1
```

```
## [1] 0.2333102
```

We see that by just shorting, we can make a lot more money. This algorithm relies on microtransactions: each

time the predicted future volatility is negative, we short and hold that position until we predict the volatility to go up again. Overall, we would make $610,700 over a period of 10 years using shorts and a particle filter. This yields a 23.33% return annually, which is somewhat competitive among financial institutions.

```
init = oil[1,2]
profit = 0
sellpt = 0
current = init
hold = TRUE
for (i in 1:nrow(oil)){
  if (current < fut_pred[i+1] && hold && abs(x_mean[i+1]) <= 0.035){
    profit = profit + (oil[i+1,2] - current)
    sellpt = oil[i+1, 2]
    hold = FALSE
  }
  #after repeated trials, 8% drop seems to be the best point to rebuy
  if (!hold && oil[i, 2] <= 0.92*sellpt){
    current = oil[i, 2]
    hold = TRUE
  }
}
tot = profit*1000 #minimum order of 1000 barrels to trade futures
tot
```

```
##   Future Price
## 1         9850
```

```
profit
```

```
##   Future Price
## 1         9.85
```

```
((tot + 85510)/85510)^(1/10)-1
```

```
##   Future Price
## 1   0.01096224
```

Using a buying strategy would yield $9850 over the past 10 years of oil futures trading. With an initial investment of $85,510 on the first day of trading, our model yields an 11.52% overall return on investment and an annual return of 1.09%, which is pretty bad.

In conclusion, taking a short position in oil futures trading, with the help of a particle filter (and in times of relative political stability) can be pretty profitable.

Bibliography

https://users.aalto.fi/~ssarkka/course_k2012/handout6.pdf

https://rpubs.com/awellis/180442 (naive code)