

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 evaluate 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:

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

Is there a reason the indices differ between the sums?

$$\begin{aligned} p(x_{0:t}|y_{1:t}) &\propto p(y_t|x_{0:t}, y_{1:t-1})p(x_{0:t}|y_{1:t-1}) \\ &= p(y_t|x_t)p(x_t|x_{0:t-1}, y_{1:t-1})p(x_{0:t-1}|y_{1:t-1}) \\ &= p(y_t|x_t)p(x_t|x_{t-1})p(x_{0:t-1}|y_{1:t-1}) \end{aligned}$$

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 $w_t^{*(i)}$:

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

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 <- 1000
```

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

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

x[, 1] <- rnorm(N, 0, sx) #Assume a normal prior

weights[, 1] <- dnorm(obs[1], x[, 1], sy) # 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] <- rnorm(N, x[, t - 1], sx) #sample x from previous x

  weights[, t] <- dnorm(obs[t], x[, t], sy) #get weights from observation
  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(aes(y = fut_pred, x = seq(1:length(fut_pred)), colour = "red"))

error <- abs(futures - fut_pred) / futures

# Absolute error as percentage of price

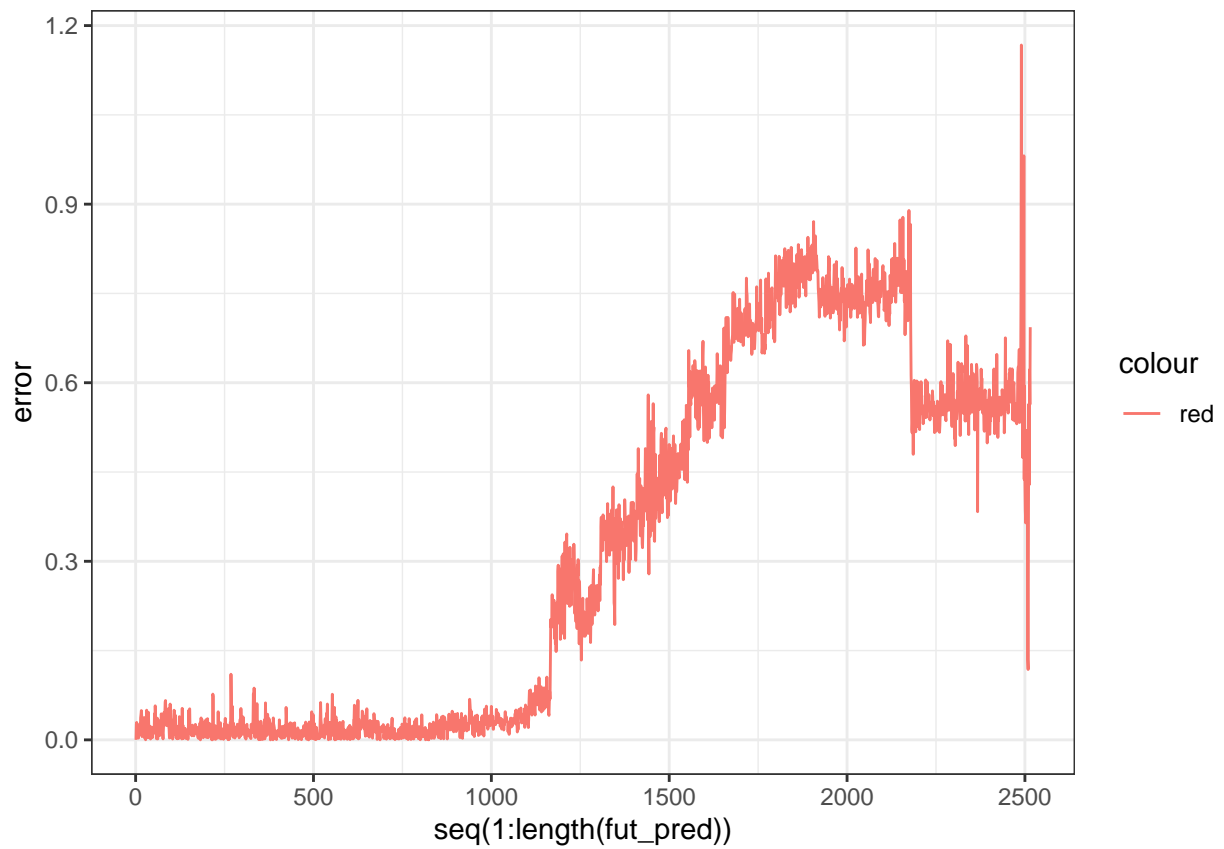
p2 <-
  ggplot() + geom_line(aes(
    y = error,
    x = seq(1:length(fut_pred)),
    colour = "red"
  ))

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

```

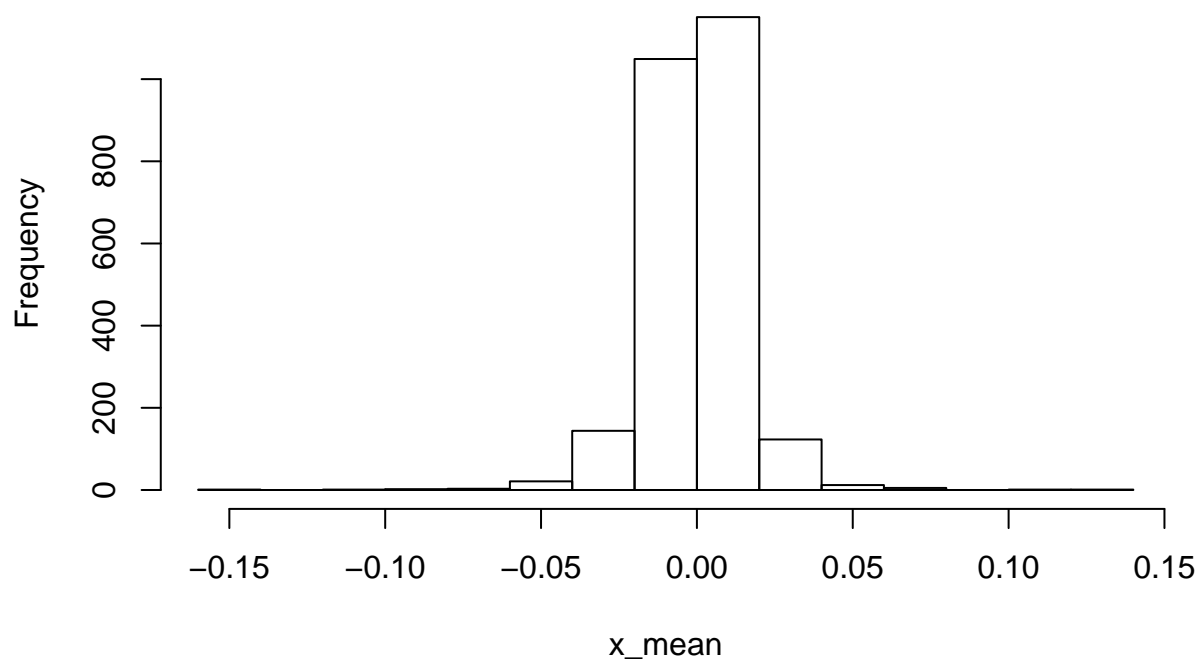


p2



```
hist(x_mean)
```

Histogram of x_mean



Bibliography

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

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