Crude Oil Price Predictor 2020

Christopher Page

8/12/2020

Introduction/Overview/Executive Summary

Oil is an important commodity. Some of our planet's inhabitants produce it. Many more consume it. All are affected by it. Because of its wide-ranging effects, oil commands collective attention enabled by analysis. One of the key metrics to take into account in the analysis of the oil indutry is the daily closing price of crude oil futures traded on a global basis in such venues as the New York Mercantile Exchange.

Methods for predicting that price may entail studying the quantitative and qualitative factors at play inside the oil industry as well as in such adjacent industries as transportation and manufacturing. They may also entail studying the often complex diplomatic, military, economic, cultural, and environmental developments that help explain why crude oil closing prices rise and fall as they do over time.

This project proposes a simpler method, one that predicts crude oil closing prices based on time and the closing prices of complementary (e.g., gasoline) and competing (e.g., platinum) commodities. The premise is that there is much to be learned by studying the behaviors of commodity traders engaged in making daily investment decisions based on running evaluations of risks and rewards.

That premise informed the development of algorithms with the potential to predict crude oil closing prices with a viable level of accuracy, quantified as the Root Means Square Error (RMSE). Using a 25,190-row data set constructed with publicly available information downloaded for free from https://www.nasdaq.com/, the author developed and then evaluated fifteen such algorithms.

Evaluation led to the selection of one algorithm that generated the lowest RMSE when applied to a test set. That algorithm used a Ranger model which took into account six key predictors:

(1) Year
(2) Month
(3) Closing Price of Heating Oil
(4) Closing Price of Gasoline
(5) Closing Price of Platinum
(6) Closing Price of Soybeans

The RMSE in question was 2.93, a figure equating to a mere 12% of the 23.51 generated by an algorithm that only took into account the average closing price across the period of observation.

Method/Analysis

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Loading required package: caTools

## Warning: package 'caTools' was built under R version 4.0.2

## Loading required package: dplyr

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

## Loading required package: dslabs

## Loading required package: forcats

## Loading required package: foreach

## Loading required package: gam

## Warning: package 'gam' was built under R version 4.0.2

## Loading required package: splines

## Loaded gam 1.20

## Loading required package: ggrepel

## Loading required package: ggthemes

## Loading required package: lubridate

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

## Loading required package: purrr

##
## Attaching package: 'purrr'
```

```
## The following objects are masked from 'package:foreach':
##
##     accumulate, when

## The following object is masked from 'package:caret':
##
##     lift

## Loading required package: randomForest

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin

## Loading required package: tidyverse

## -- Attaching packages ---------------------------------------------------------------------------

## v tibble  3.0.1     v readr   1.3.1
## v tidyr   1.1.0     v stringr 1.4.0

## -- Conflicts ------------------------------------------------------------------------------------
## x purrr::accumulate()      masks foreach::accumulate()
## x lubridate::as.difftime() masks base::as.difftime()
## x randomForest::combine()  masks dplyr::combine()
## x lubridate::date()        masks base::date()
## x dplyr::filter()          masks stats::filter()
## x lubridate::intersect()   masks base::intersect()
## x dplyr::lag()             masks stats::lag()
## x purrr::lift()            masks caret::lift()
## x randomForest::margin()   masks ggplot2::margin()
## x lubridate::setdiff()     masks base::setdiff()
## x lubridate::union()       masks base::union()
## x purrr::when()            masks foreach::when()

## Parsed with column specification:
## cols(
##   Date = col_character(),
##   Commodity = col_character(),
##   Sector = col_character(),
##   `Closing Price` = col_double()
## )
```

```
## Joining, by = "date"
## Joining, by = "date"
## Joining, by = "date"
## Joining, by = "date"
## Joining, by = "date"
## Joining, by = "date"
## Joining, by = "date"
## Joining, by = "date"
## Joining, by = "date"
```

```
## [1] "Crude Oil Commodity Prices began the August 2010 to July 2020 period at"
```

```
## [1] 78.95
```

```
## [1] "They ended that ten-year period at"
```

```
## [1] 40.34
```

```
## [1] "During that time, they averaged"
```

```
## [1] 70.13307
```

```
## [1] "Running from a minimum of"
```

```
## [1] -37.25
```

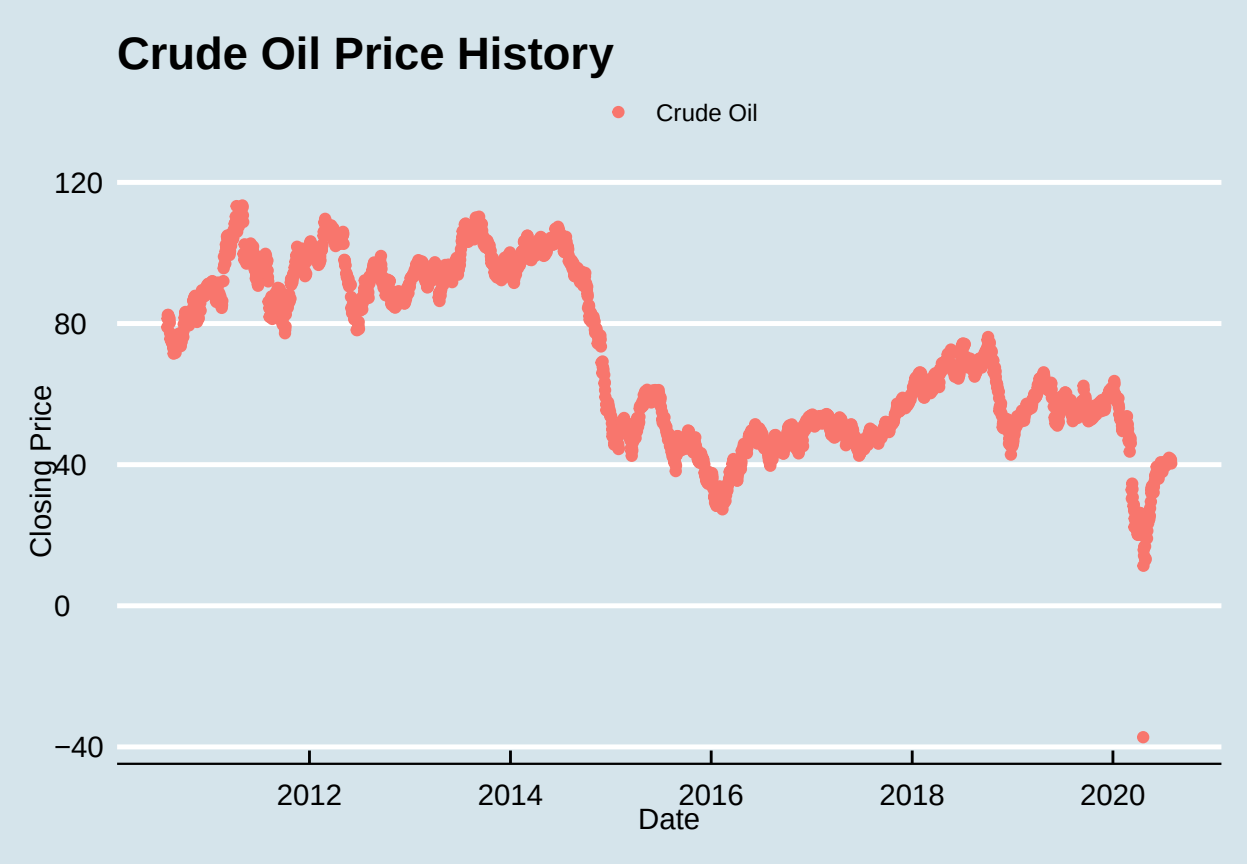
```
## [1] "To a maximum of"
```

```
## [1] 113.45
```

```
## [1] "With a standard deviation of"
```

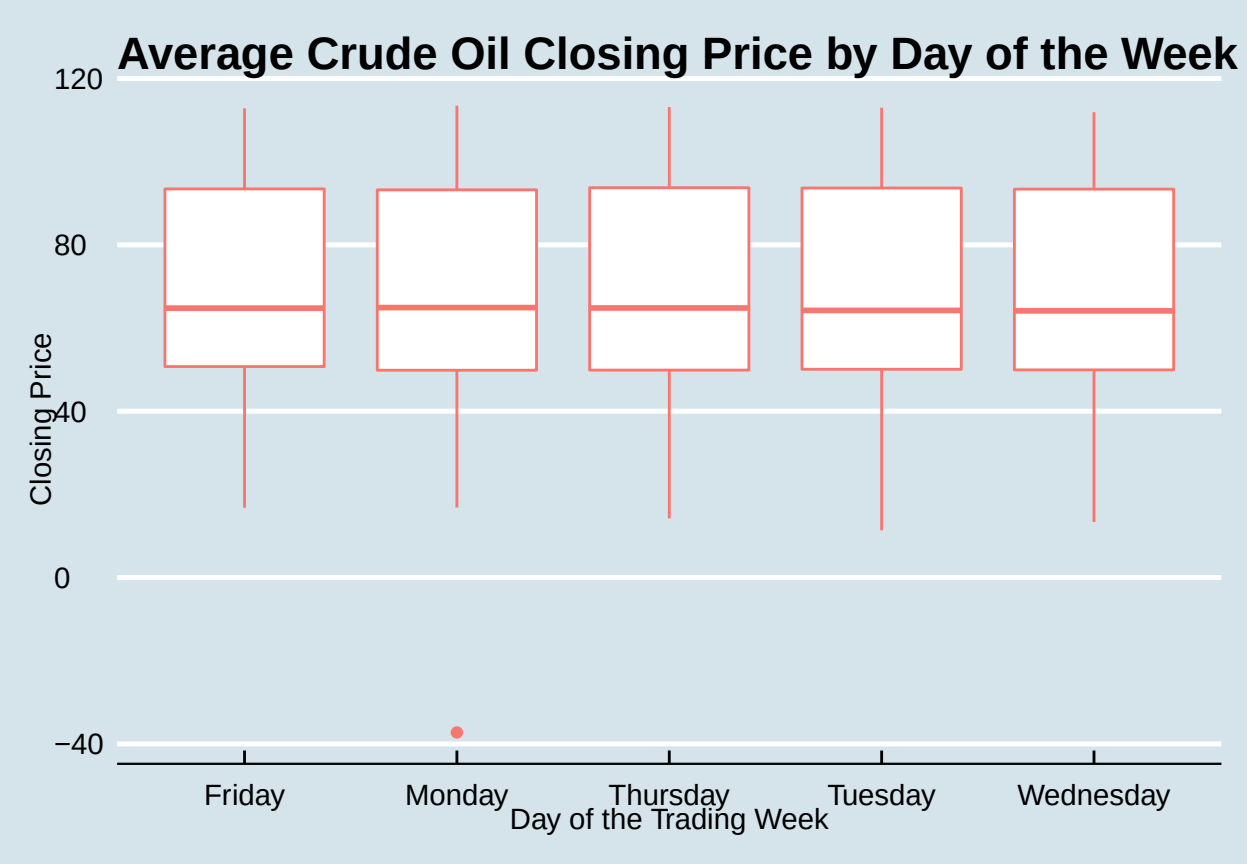
```
## [1] 23.52209
```

```
### Second, this script plots the 2010-2020 timeline of crude oil daily closing prices
### Then, it saves the plot in a .png file entitled "crude_oil_price_history"

crude_oil_plot <- crude_oil %>%
  ggplot(aes(date, closing_price)) +
  geom_point(aes(color = commodity)) +
  xlab("Date") +
  ylab("Closing Price") +
  ggtitle("Crude Oil Price History") +
  theme_economist() +
    theme(legend.position="top",
          legend.title = element_blank(),
          legend.box = "horizontal" ,
          legend.text=element_text(size=8.5)) +
    guides(col = guide_legend(nrow = 1))

crude_oil_plot
```

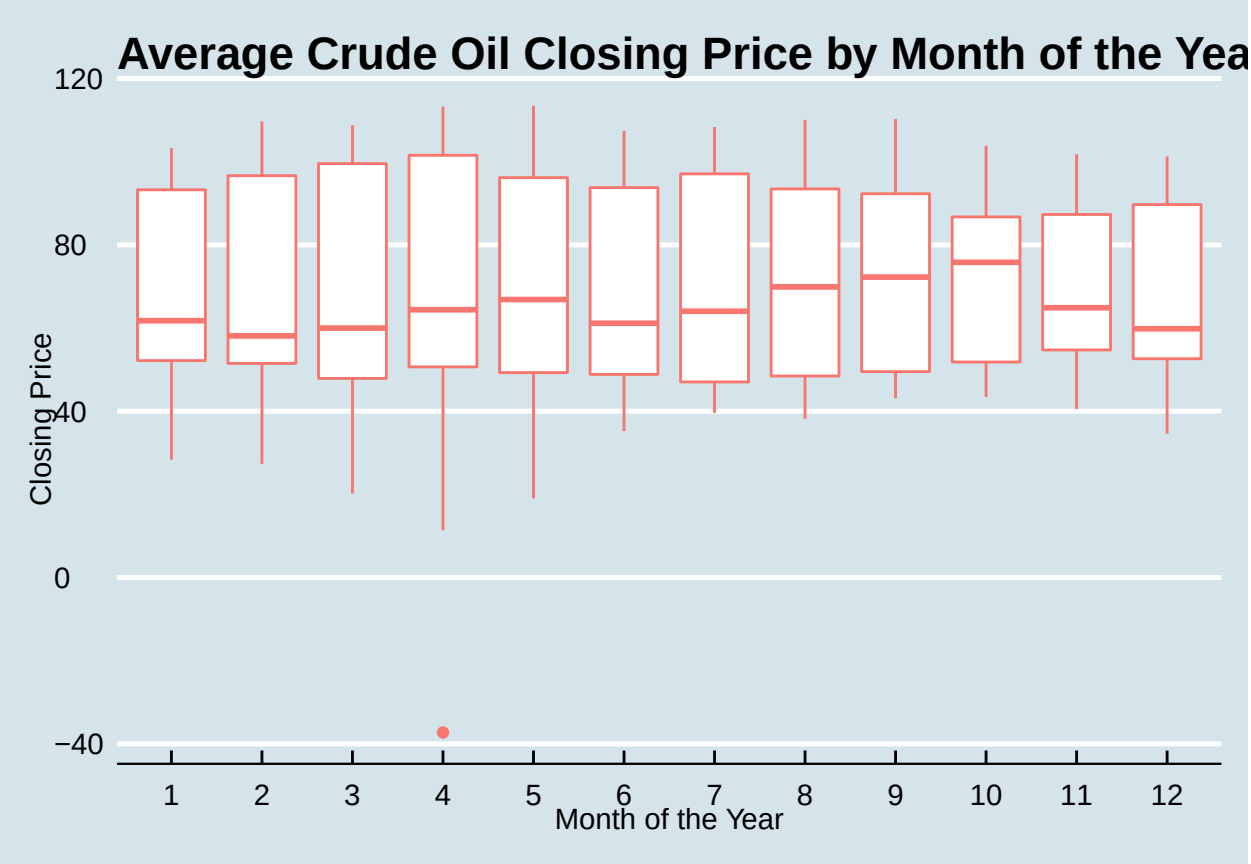



```
ggsave("fig/crude_oil_price_history.png")
```

```
## Saving 6.5 x 4.5 in image
```

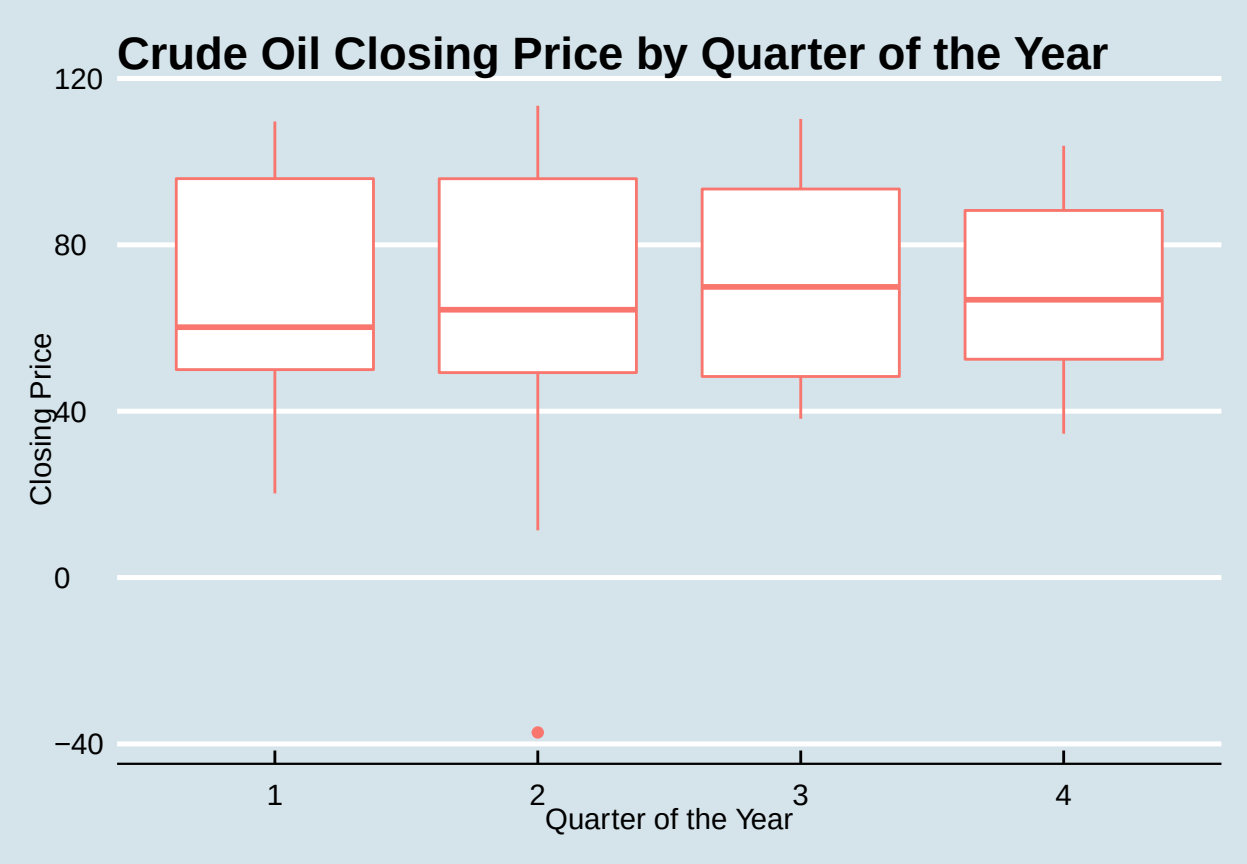



```
## Saving 6.5 x 4.5 in image
```



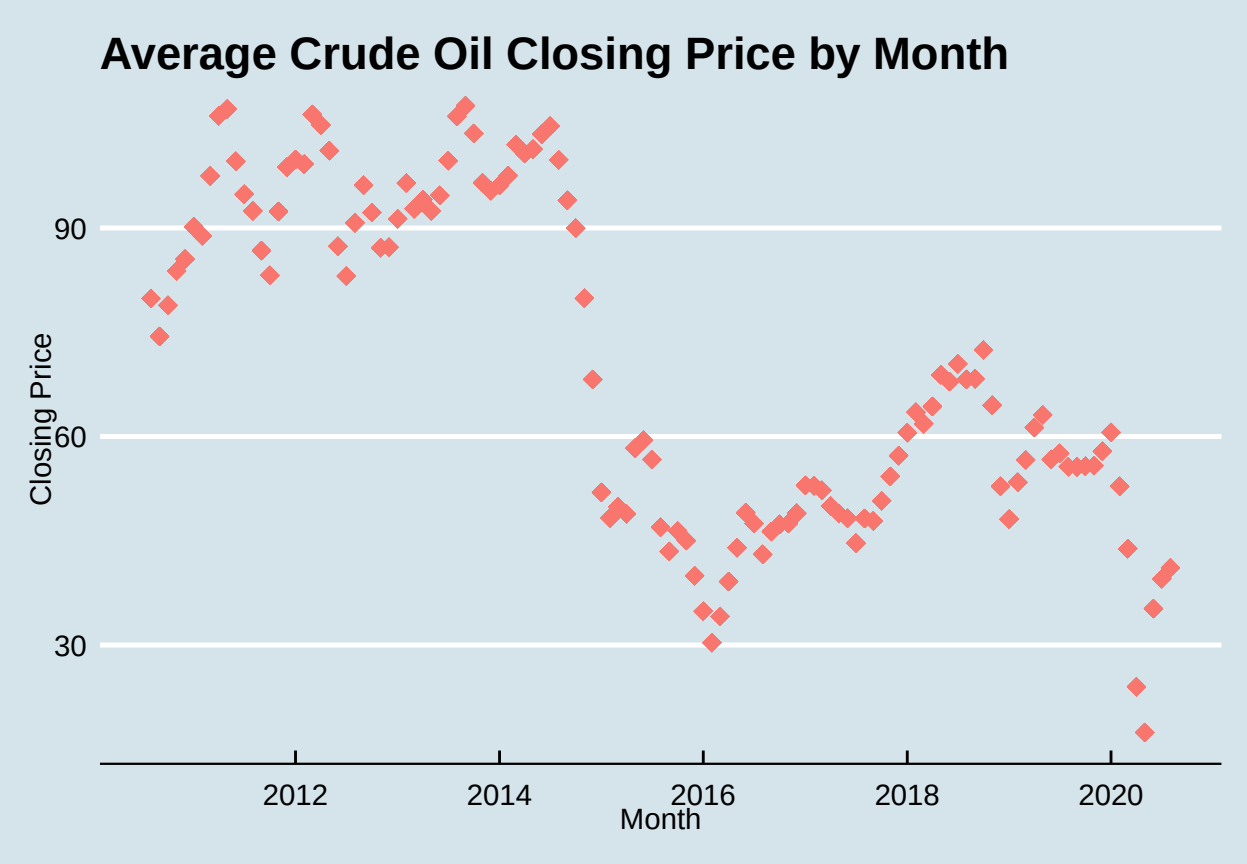

```
## Saving 6.5 x 4.5 in image
```

```
crude_oil_average_by_quarter_of_the_year_plot
```

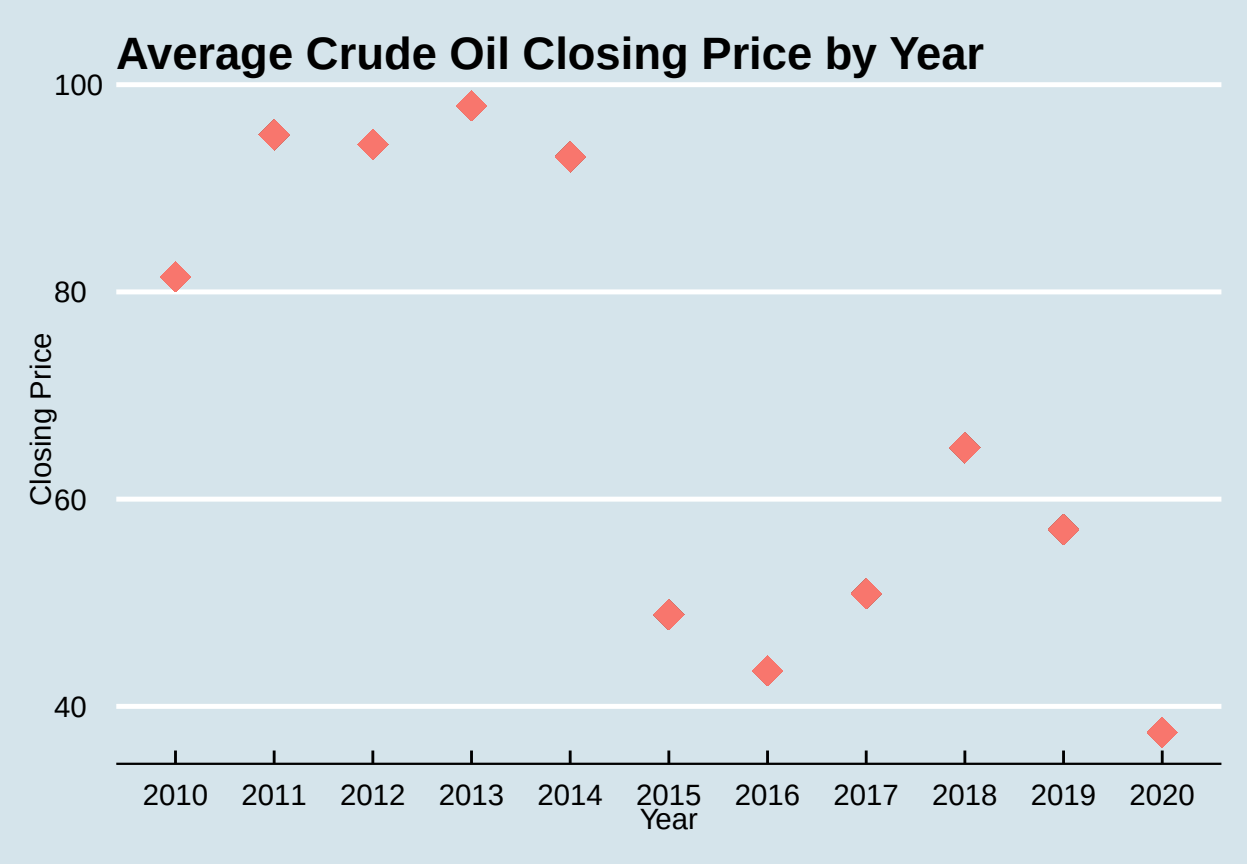



```
ggsave("fig/crude_oil_average_by_quarter_of_the_year.png")
```

```
## Saving 6.5 x 4.5 in image
```

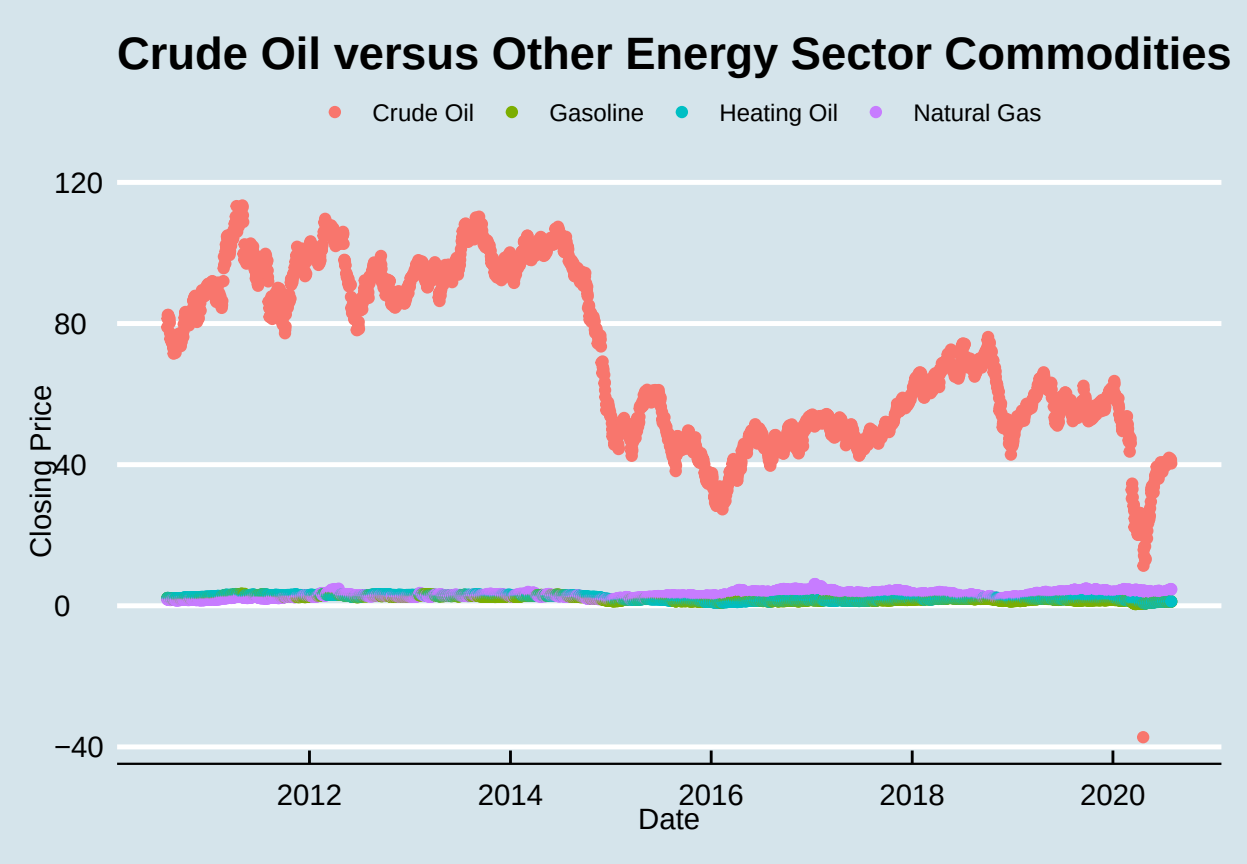



```
## Saving 6.5 x 4.5 in image
```



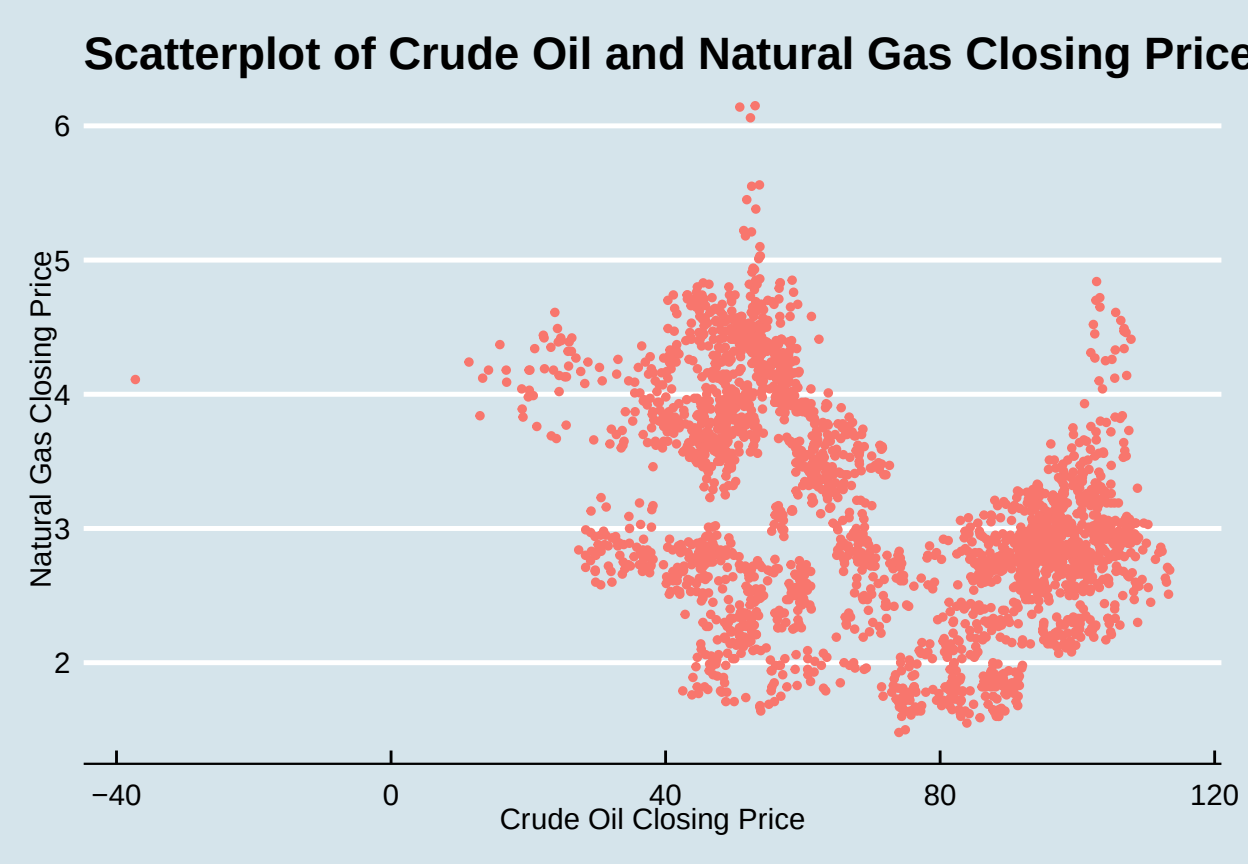

```
## Saving 6.5 x 4.5 in image
```

```
crude_oil_vs_other_energy_plot
```



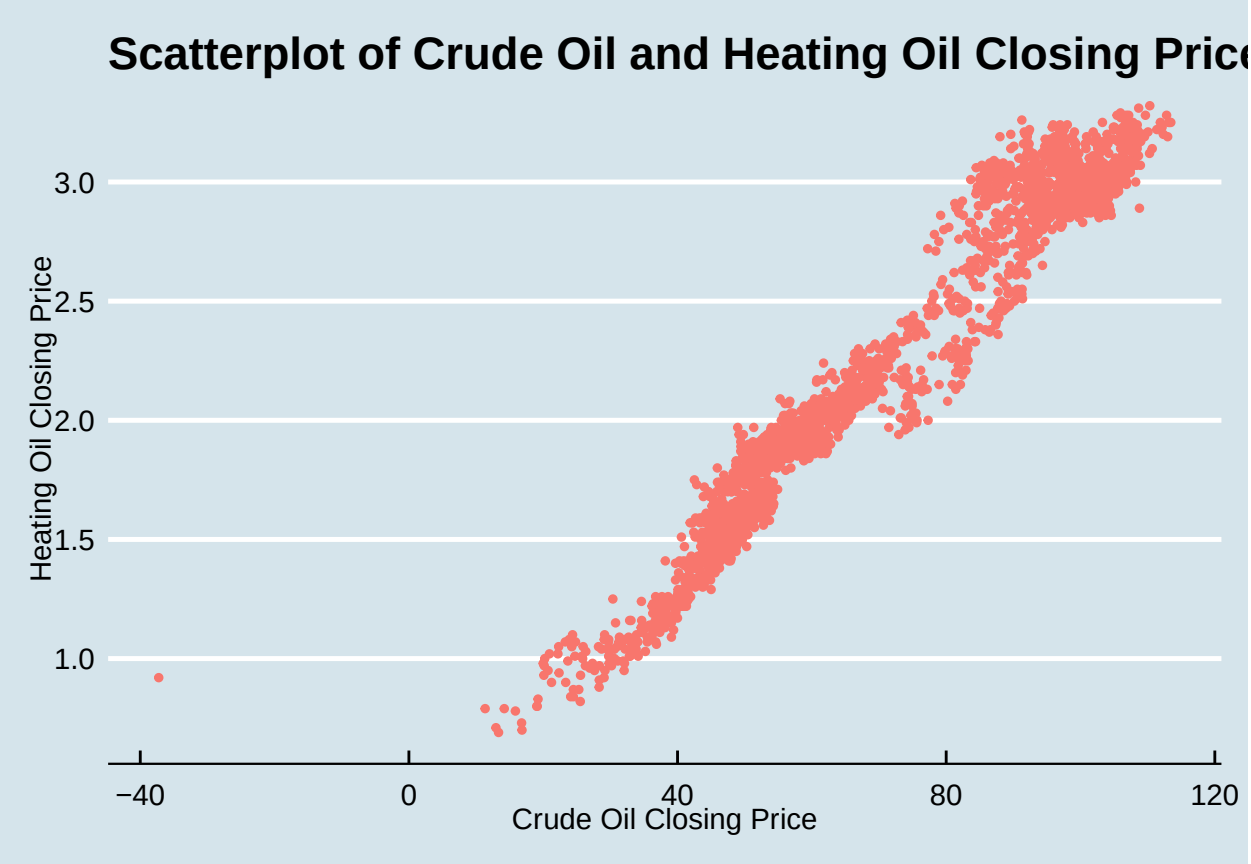

```
ggsave("fig/crude_oil_in_comparison_to_three_other_energy_sector_commodities.png")
```

```
## Saving 6.5 x 4.5 in image
```



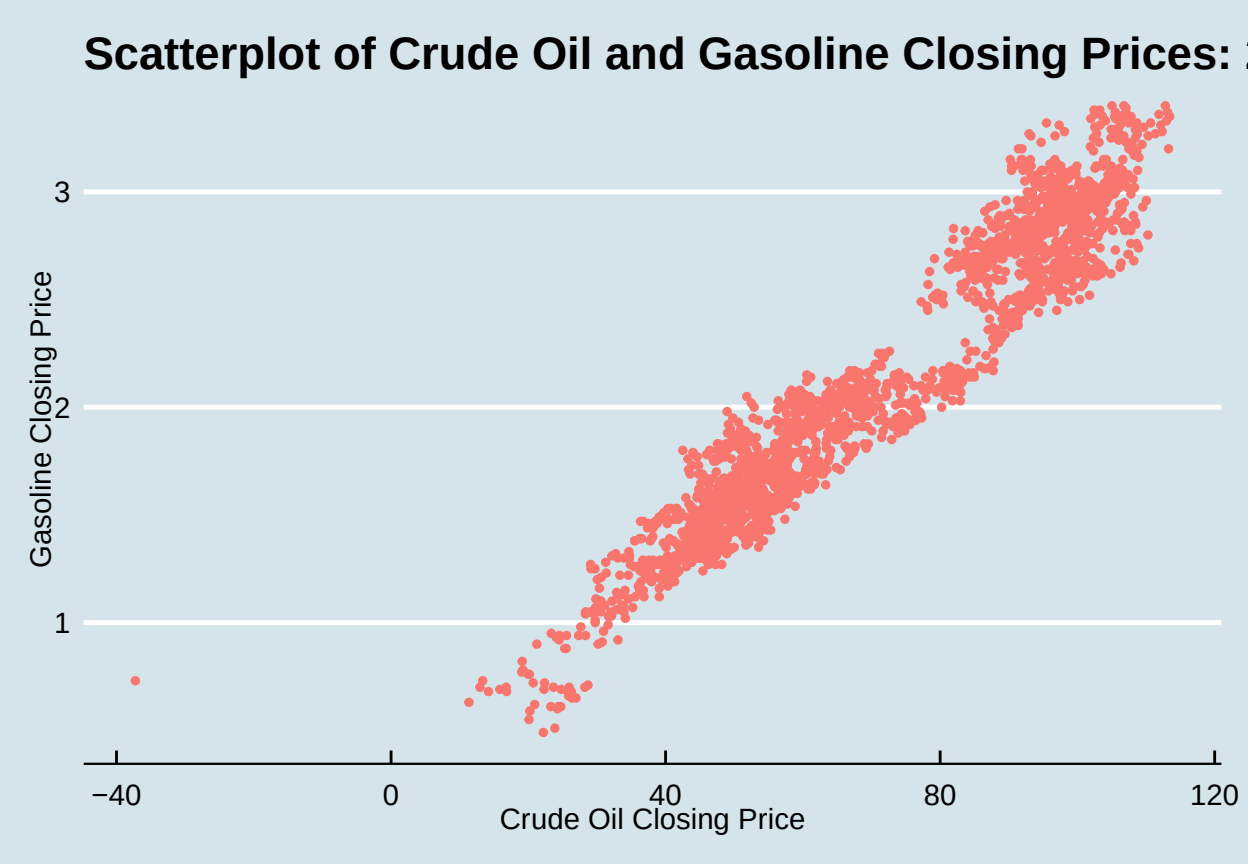

```
## Saving 6.5 x 4.5 in image
```

```
crude_oil_heating_oil_scatterplot
```

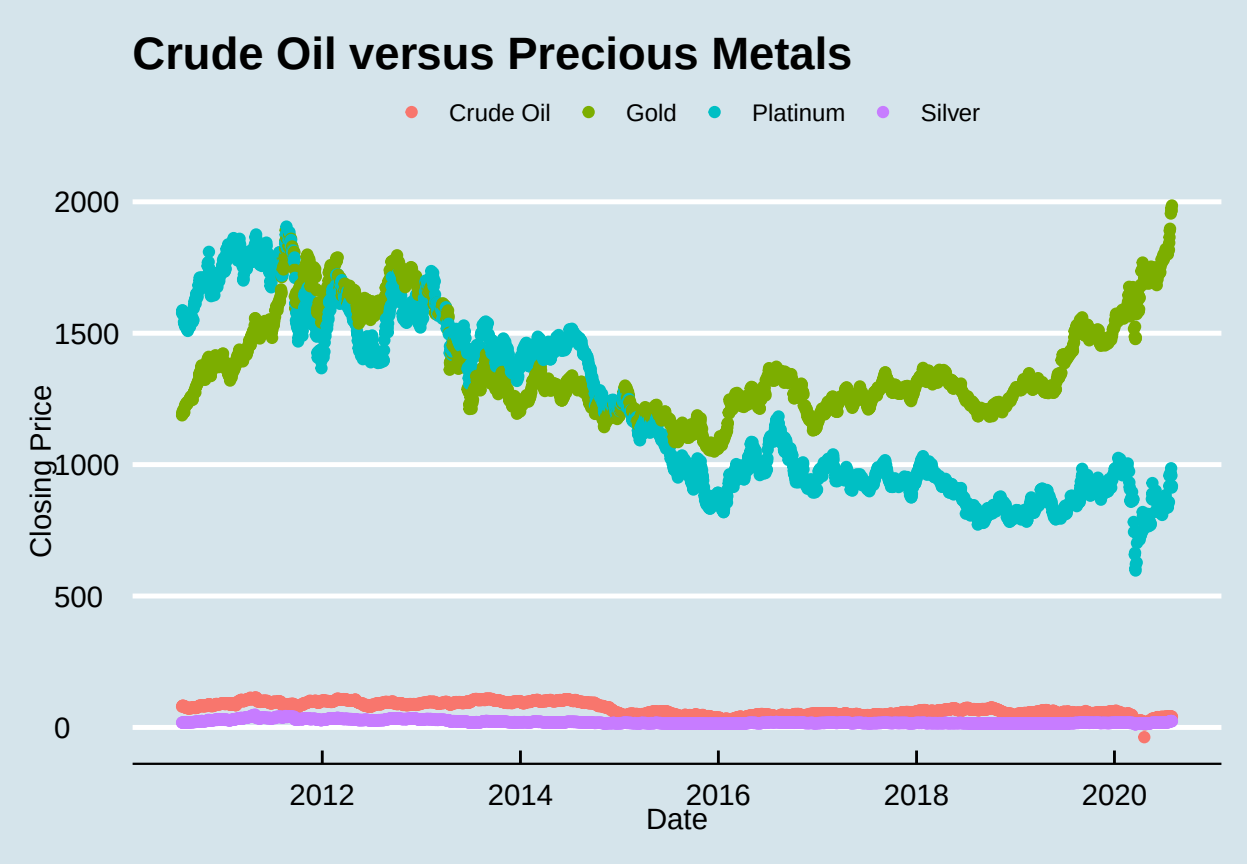



```
ggsave("fig/scatterplot_crude_oil_versus_heating_oil.png")
```

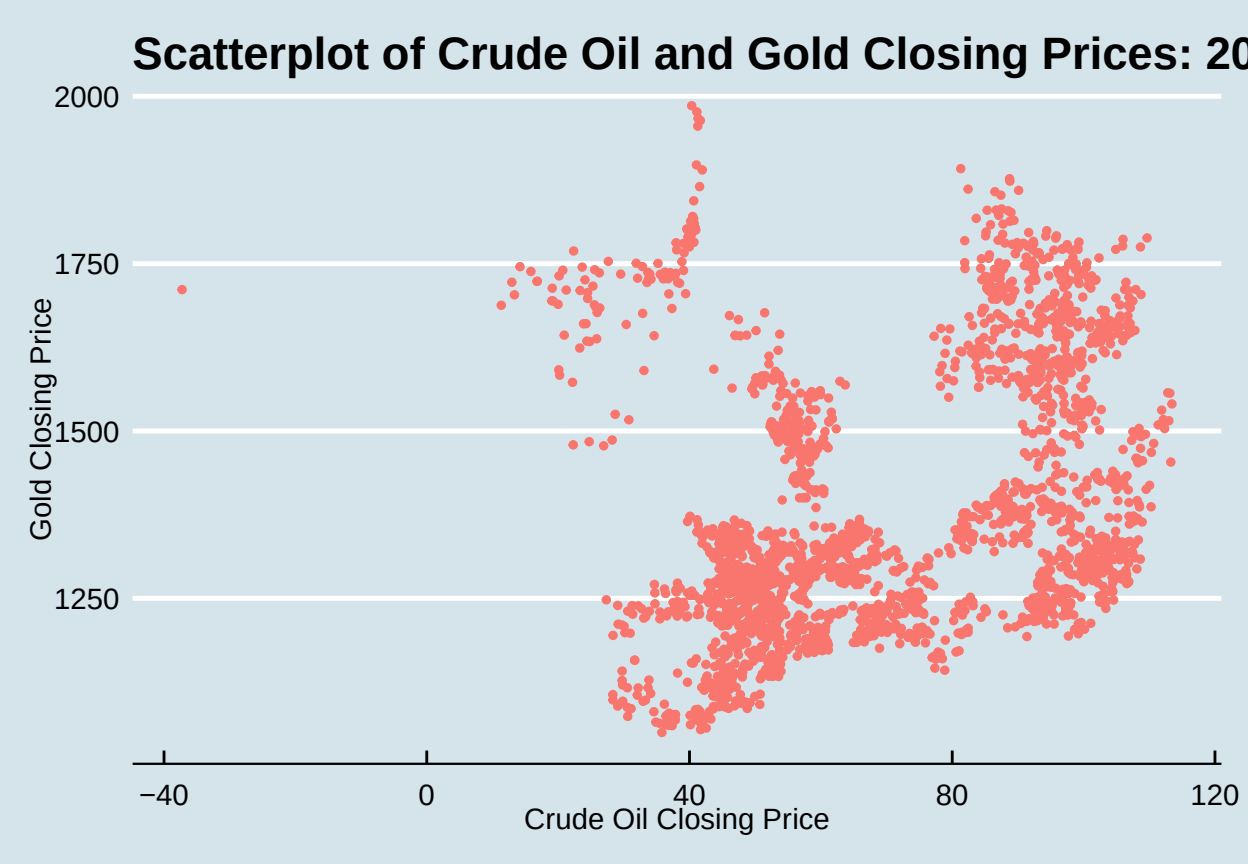
```
## Saving 6.5 x 4.5 in image
```



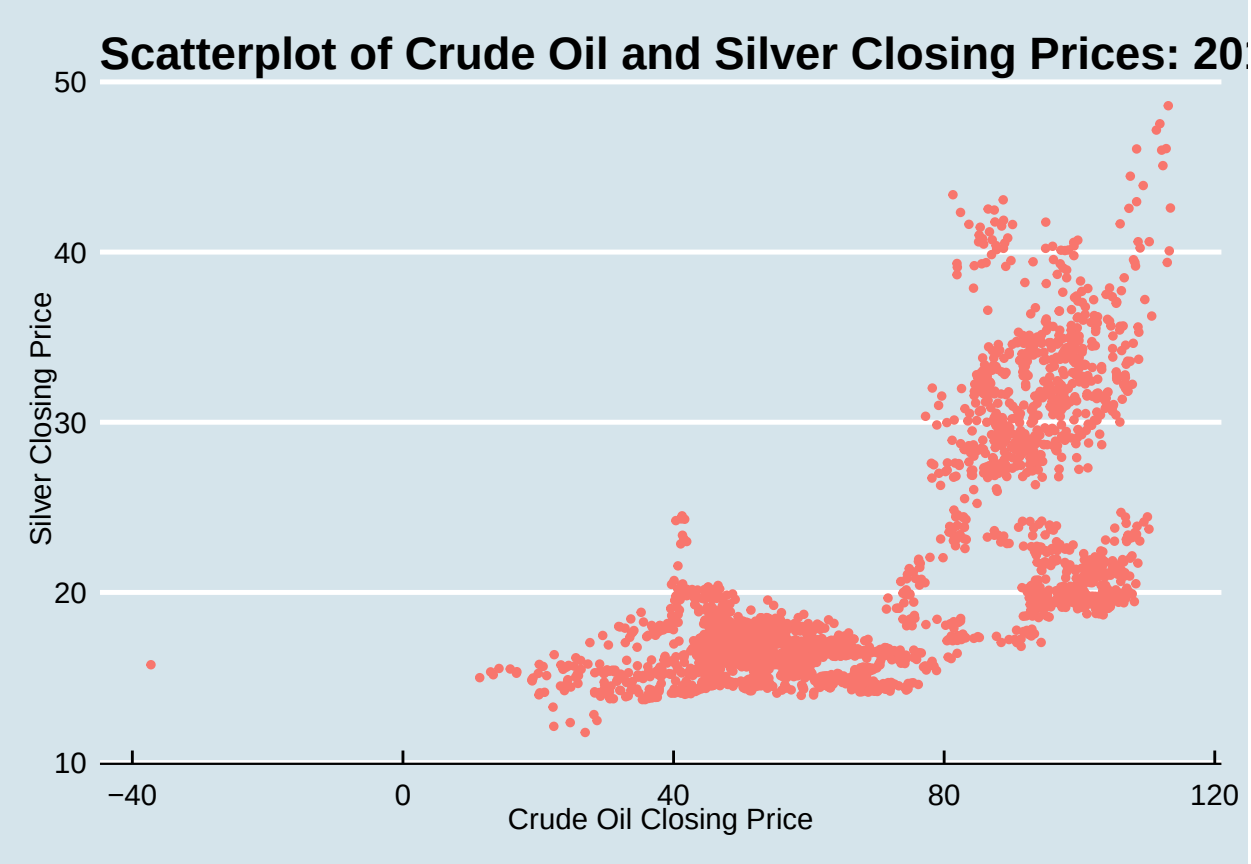

```
## Saving 6.5 x 4.5 in image
```




```
## Saving 6.5 x 4.5 in image
```

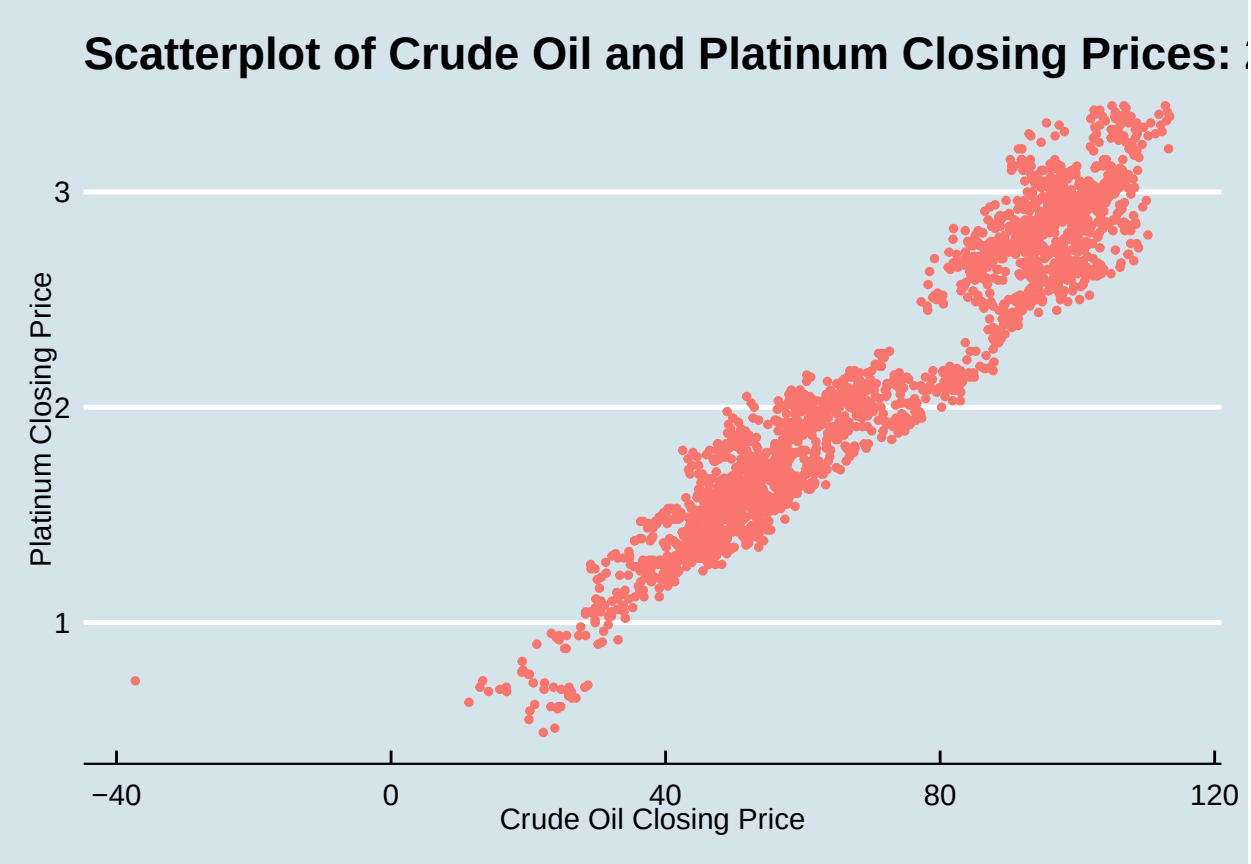



```
## Saving 6.5 x 4.5 in image
```



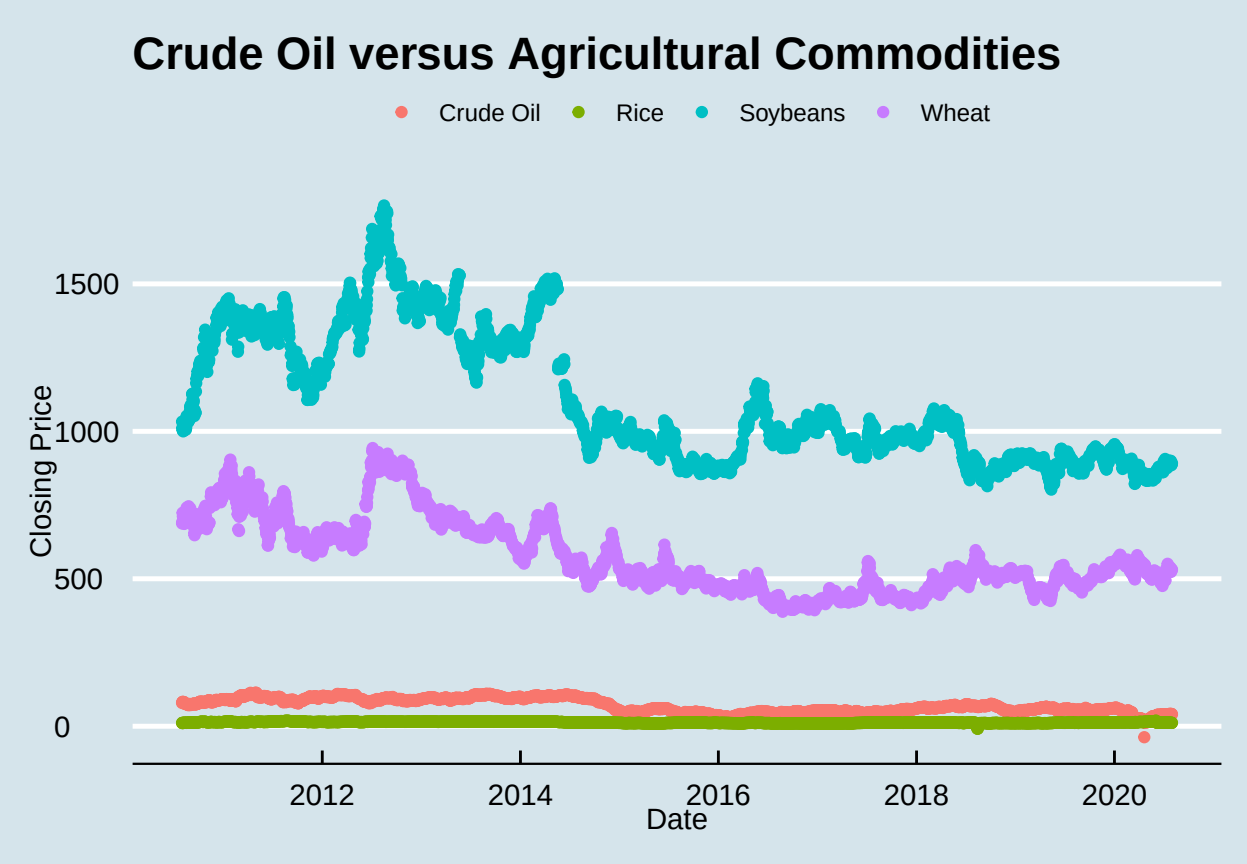

```
## Saving 6.5 x 4.5 in image
```

```
crude_oil_platinum_scatterplot
```

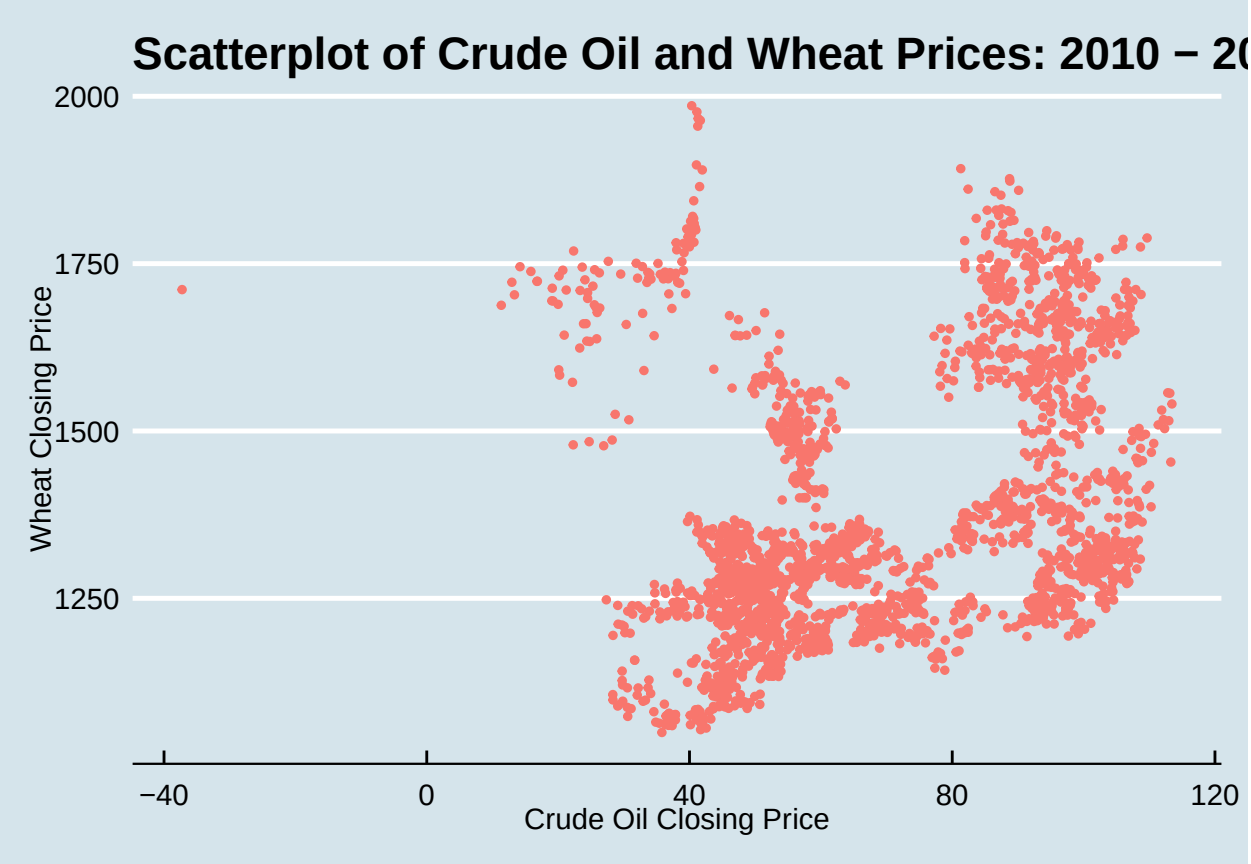



```
ggsave("fig/scatterplot_crude_oil_versus_platinum.png")
```

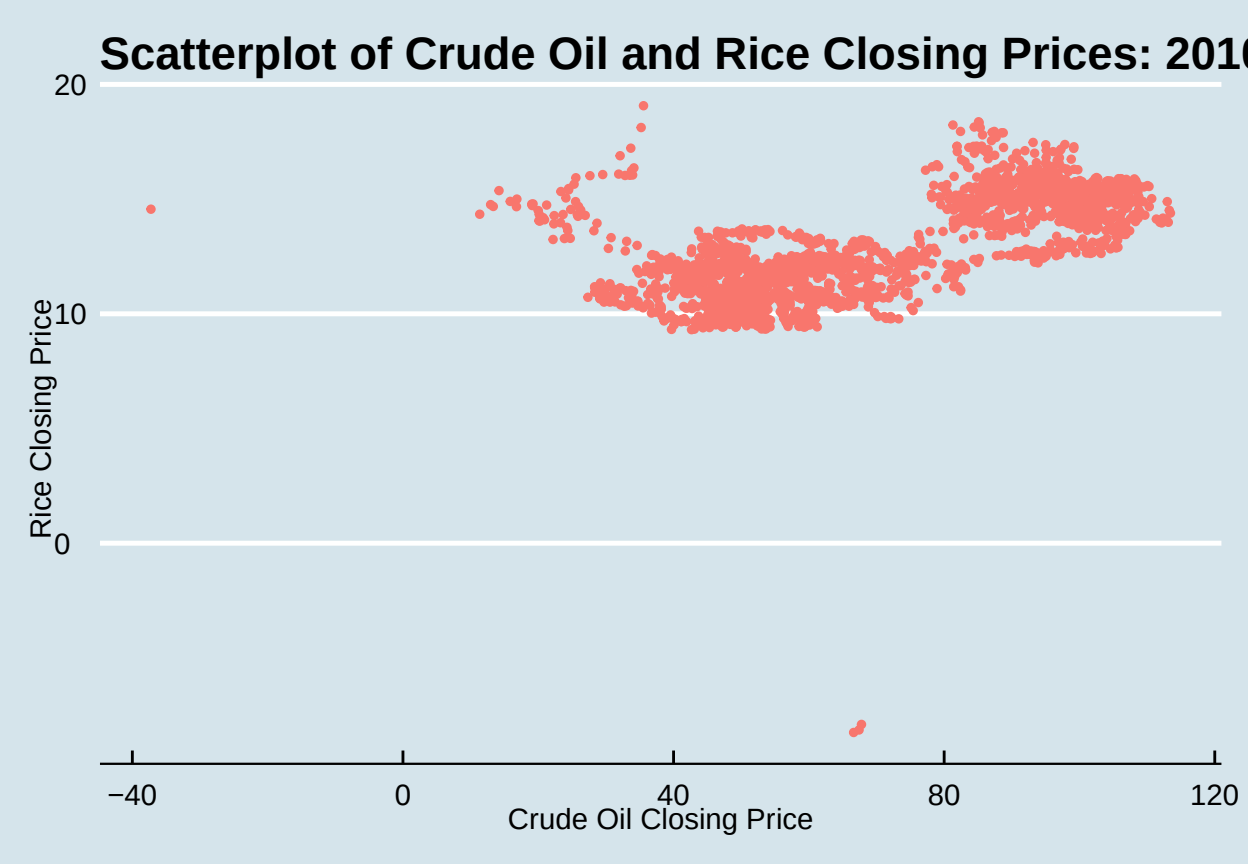
```
## Saving 6.5 x 4.5 in image
```




```
## Saving 6.5 x 4.5 in image
```

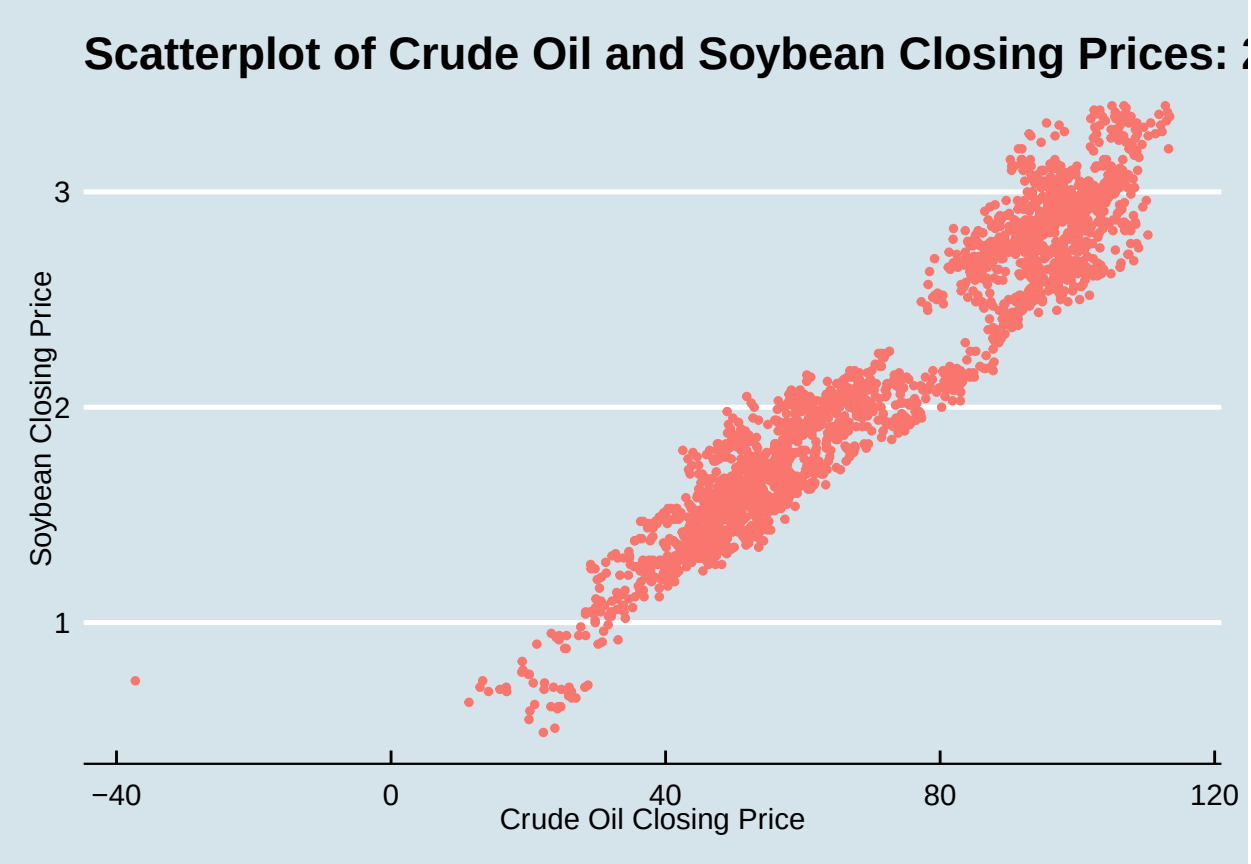



```
## Saving 6.5 x 4.5 in image
```




```
## Saving 6.5 x 4.5 in image
```

```
crude_oil_soybeans_scatterplot
```




```
ggsave("fig/scatterplot_crude_oil_versus_soybean.png")
```

```
## Saving 6.5 x 4.5 in image
```

```
## Warning: The ‘i‘ argument of ‘‘[‘()‘ can’t be a matrix as of tibble 3.0.0.
## Convert to a vector.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_warnings()‘ to see where this warning was generated.
```

```
### This section evaluates a set of fifteen potentially viable algorithms for predicting cruide oil clo

### Each of the first nine algorithms considers crude oil in isolation from other globally traded commo
### In isolation, the available predictors include both the cyclical and the linear components of time
### One cyclical component is each observation's day of the trading week (Monday, Tuesday,...Friday)
### Others are the month (1,2..12) and the quarter (1,2,..4) of each year on the 2010-2020 timeline
### Liner components are the year (e.g., 2010) and month (e.g., 8/2010) of each observation

#### ALGORITHM 01 (AVERAGE)

##### This algorithm predicts crude oil closing prices based solely on the average crude oil price for
##### It will serve as a baseline from which to compare the analytical viability of other, more sophist

mu <- mean(crude_oil_train$closing_price)

predicted_price_algorithm_01 <- mu
```

```
RMSE01 <- RMSE(predicted_price_algorithm_01, crude_oil_test$closing_price)

RMSE01
```

```
## [1] 23.51292
```

```
#### ALGORITHM 02 (WEEKDAY EFFECTS)

##### This algorithm predicts crude oil closing prices based on average crude oil price plus the effect

crude_oil_average_by_day_of_the_week <- crude_oil_train %>%
  group_by(date_weekday) %>%
  summarize(b_w = mean(closing_price))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
predicted_price_algorithm_02 <- crude_oil_test %>%
  left_join(crude_oil_average_by_day_of_the_week, by= 'date_weekday') %>%
  mutate(pred = b_w) %>%
  pull(pred)

RMSE02 <- RMSE(predicted_price_algorithm_02, crude_oil_test$closing_price)

RMSE02
```

```
## [1] 23.51689
```

```
#### ALGORITHM 03 (MONTH OF THE YEAR EFFECTS)
##### This algorithm predicts crude oil closing prices based on average crude oil price plus the effect

crude_oil_average_by_month <- crude_oil_train %>%
  group_by(date_month_of_the_year) %>%
  summarize(b_m_y = mean(closing_price))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
predicted_price_algorithm_03 <- crude_oil_test %>%
  left_join(crude_oil_average_by_month, by= 'date_month_of_the_year') %>%
  mutate(pred = b_m_y) %>%
  pull(pred)

RMSE03 <- RMSE(predicted_price_algorithm_03, crude_oil_test$closing_price)

RMSE03
```

```
## [1] 23.54449
```

```
#### ALGORITHM 04 (QUARTER OF THE YEAR EFFECTS)
##### This algorithm predicts crude oil closing prices based on average crude oil price plus the effect

crude_oil_average_by_quarter_of_the_year <- crude_oil_train %>%
  group_by(date_quarter_of_the_year) %>%
  summarize(b_q_y = mean(closing_price))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
predicted_price_algorithm_04 <- crude_oil_test %>%
  left_join(crude_oil_average_by_quarter_of_the_year, by= 'date_quarter_of_the_year') %>%
  mutate(pred = b_q_y) %>%
  pull(pred)

RMSE04 <- RMSE(predicted_price_algorithm_04, crude_oil_test$closing_price)

RMSE04
```

```
## [1] 23.51713
```

```
#### ALGORITHM 05 (YEAR EFFECTS)
##### This algorithm predicts crude oil closing prices based on average crude oil price plus the effect

crude_oil_average_by_year <- crude_oil_train %>%
  group_by(date_year) %>%
  summarize(b_y = mean(closing_price))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
predicted_price_algorithm_05 <- crude_oil_test %>%
  left_join(crude_oil_average_by_year, by= 'date_year') %>%
  pull(b_y)

RMSE05 <- RMSE(predicted_price_algorithm_05, crude_oil_test$closing_price)

RMSE05
```

```
## [1] 8.268093
```

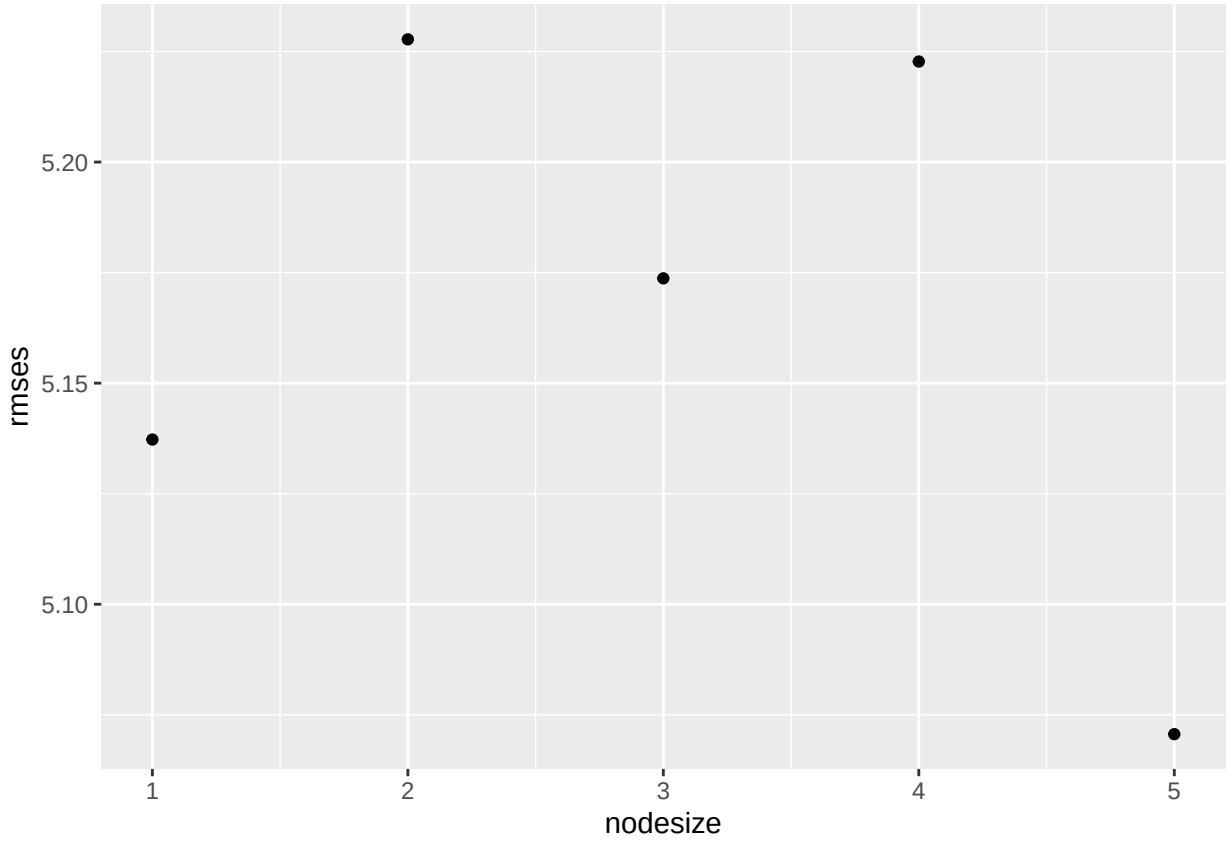
```
#### ALGORITHM 06 (MONTH EFFECTS)
##### This algorithm predicts crude oil closing prices based on average crude oil price plus the effect

crude_oil_average_by_month <- crude_oil_train %>%
  group_by(date_month) %>%
  summarize(b_m = mean(closing_price))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
predicted_price_algorithm_06 <- crude_oil_test %>%
  left_join(crude_oil_average_by_month, by= 'date_month') %>%
  mutate(pred = b_m) %>%
  pull(pred)

RMSE06 <- RMSE(predicted_price_algorithm_06, crude_oil_test$closing_price)

RMSE06
```

```
## [1] 3.800524
```

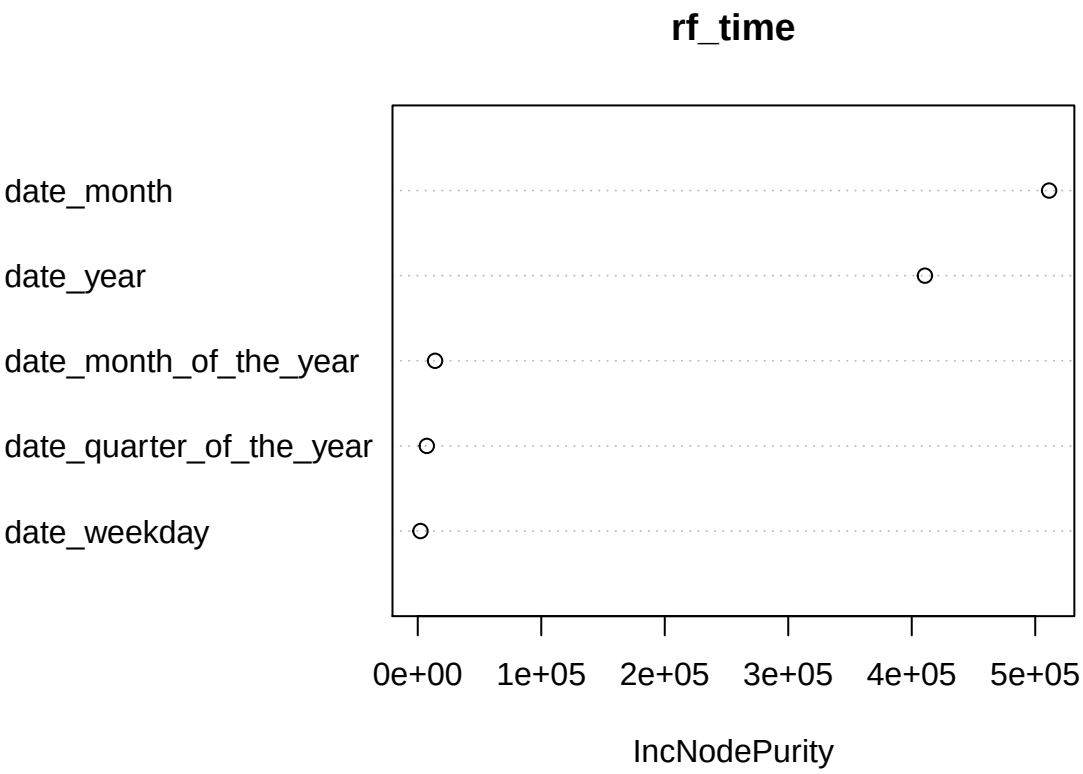
```r
#### ALGORITHM 07 (RANDOM FOREST - TIME)
##### This algorithm predicts crude oil closing prices based on a Random Forest of time components
####### The first step is to look at a Random Forest incorporating all of above-listed components of ti

nodesize <- seq(1, 5, 1)

rmses <- sapply(nodesize, function(n){
  rf_time = randomForest(closing_price ~
                           date_weekday +
                           date_month_of_the_year +
                           date_quarter_of_the_year +
                           date_year +
                           date_month,
                         data = crude_oil_train, nodesize = n)
  pred <- predict(rf_time, newdata = crude_oil_train)
  RMSE(pred, crude_oil_train$closing_price)
})

qplot(nodesize, rmses)
```



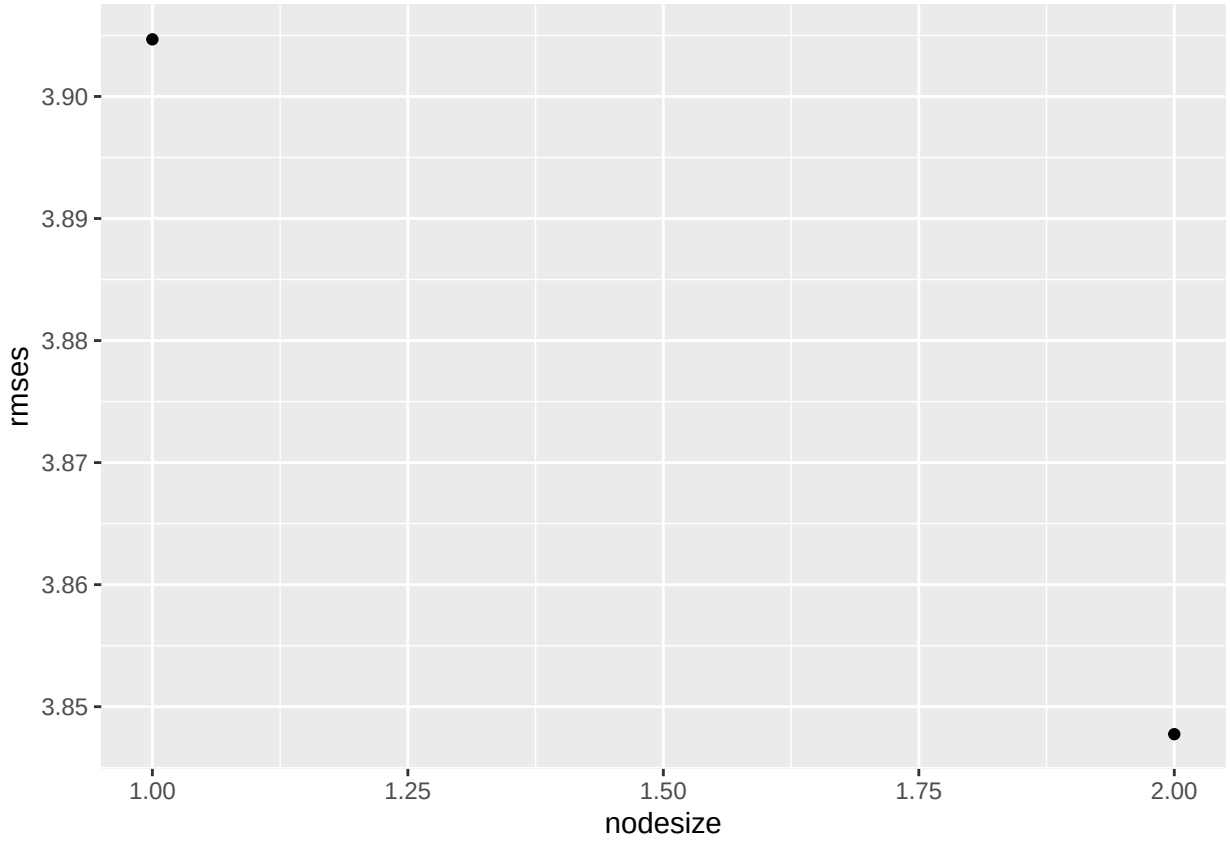

```
nodesize[which.min(rmses)]
```

```
## [1] 5
```

```
rf_time = randomForest(closing_price ~
                         date_weekday +
                         date_month_of_the_year +
                         date_quarter_of_the_year +
                         date_year +
                         date_month,
                       data = crude_oil_train, nodesize = nodesize[which.min(rmses)])

####### The second is to build a plot to help determine which of those components is/are the most impor

varImpPlot(rf_time)
```




```
####### Based on that plot, Random Forest will be revised to focus on the month and year

nodesize <- seq(1, 2, 1)

rmses <- sapply(nodesize, function(n){
  rf_time = randomForest(closing_price ~
                           date_month +
                           date_year,
                         data = crude_oil_train, nodesize = n)
  pred <- predict(rf_time, newdata = crude_oil_train)
  RMSE(pred, crude_oil_train$closing_price)
})

qplot(nodesize, rmses)
```




```
nodesize[which.min(rmses)]
```

```
## [1] 2
```

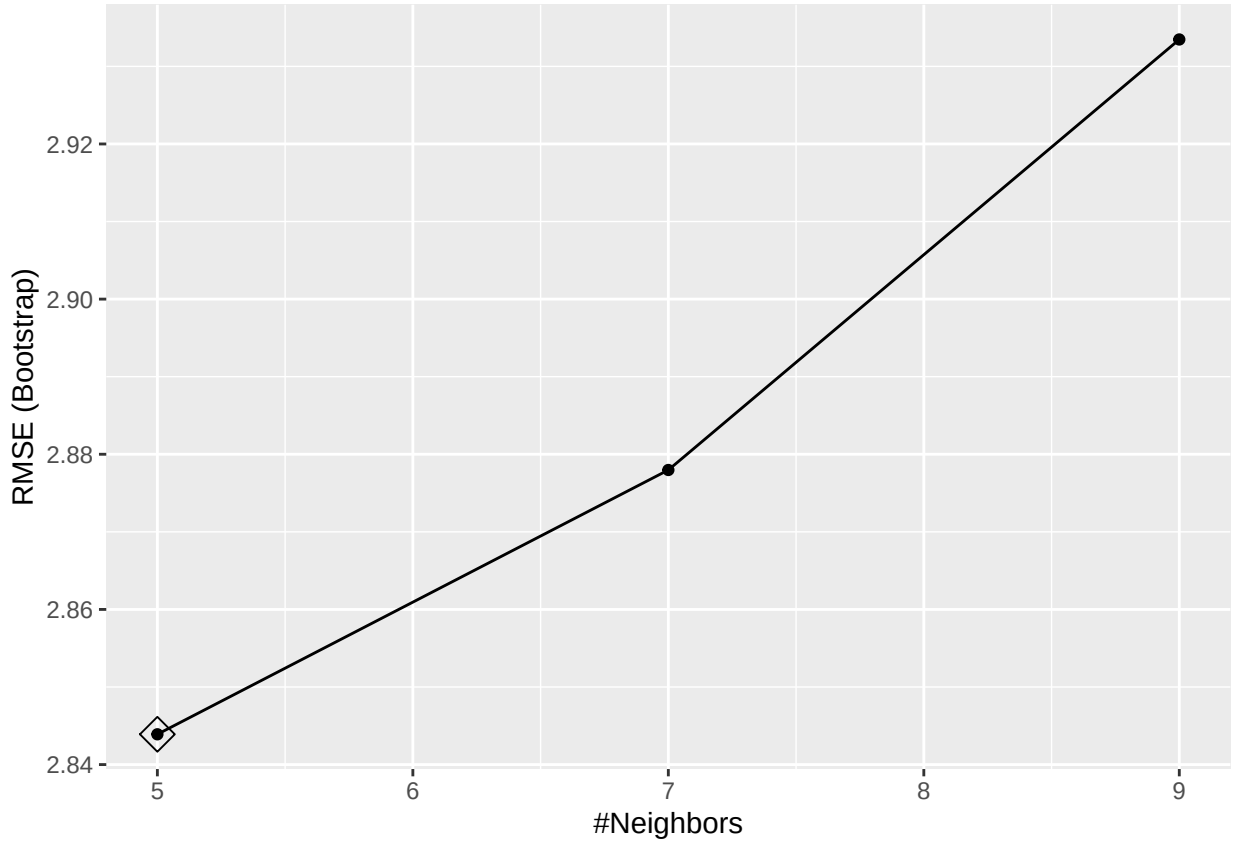
```
rf_time = randomForest(closing_price ~
                         date_month +
                         date_year,
                       data = crude_oil_train, nodesize = nodesize[which.min(rmses)])

pred <- predict(rf_time, newdata = crude_oil_train)

RMSE(pred, crude_oil_train$closing_price)
```

```
## [1] 3.850352
```

```
predicted_price_algorithm_07 <- predict(rf_time, newdata = crude_oil_test)

RMSE07 <- RMSE(predicted_price_algorithm_07, crude_oil_test$closing_price)

RMSE07
```

```
## [1] 4.716089
```

```
#### ALGORITHM 08 (KNN - TIME)
##### This algorithm predicts crude oil closing prices based on a K-Nearest Neighbors (KNN) model of th

knn_time <- train(closing_price ~
                    date_year +
                    date_month,
                  method = "knn",
                  data = crude_oil_train)

ggplot(knn_time, highlight = TRUE)
```




```
pred <- predict(knn_time, newdata = crude_oil_train)

RMSE(pred, crude_oil_train$closing_price)
```

```
## [1] 2.572097
```

```
predicted_price_algorithm_08 <- predict(knn_time, newdata = crude_oil_test)

RMSE08 <- RMSE(predicted_price_algorithm_08, crude_oil_test$closing_price)

RMSE08
```

```
## [1] 3.768831
```

```
#### ALGORITHM 09 (RANGER - TIME)
##### This algorithm predicts crude oil closing prices based on a Ranger model of the two key time comp

ranger_time <- train(closing_price ~
                       date_year +
                       date_month,
                     method = "ranger",
                     data = crude_oil_train)
```

```
## note: only 1 unique complexity parameters in default grid. Truncating the grid to 1 .
```

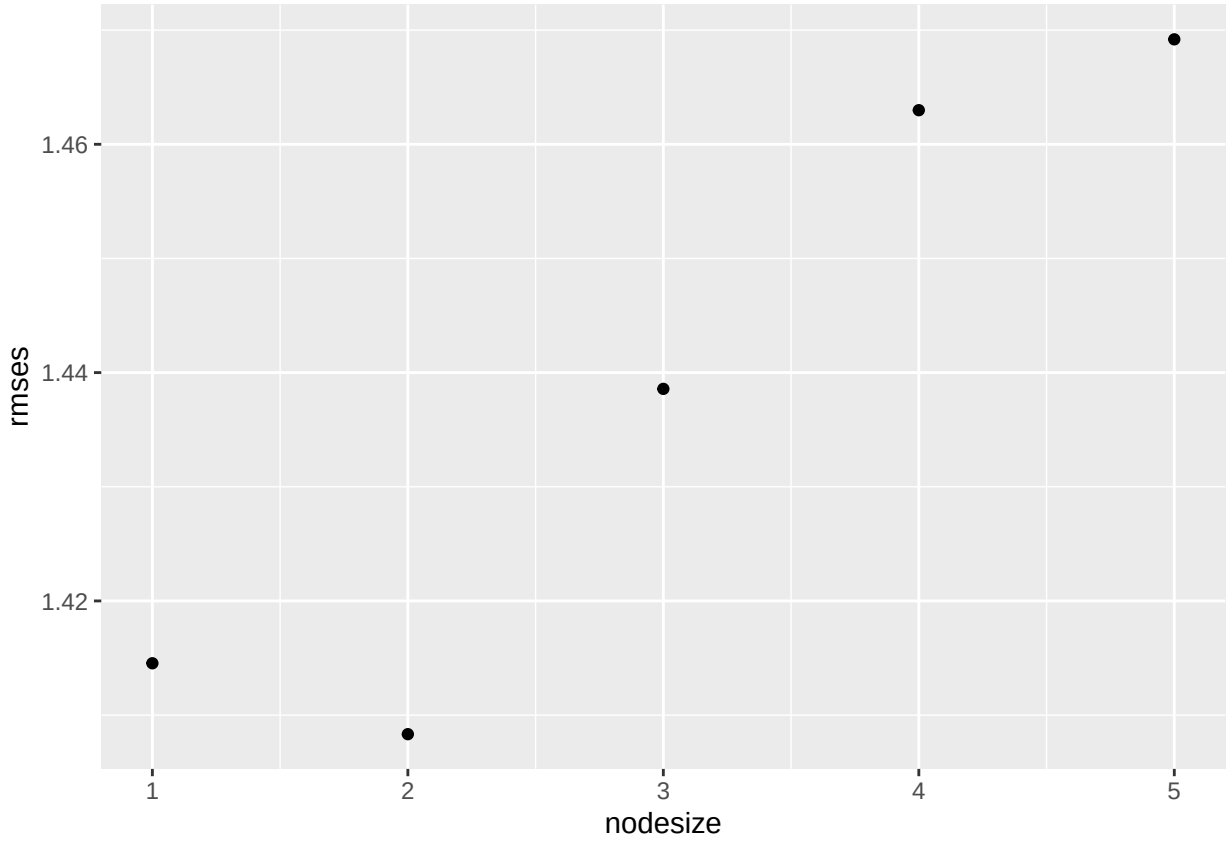
```
pred <- predict(ranger_time, newdata = crude_oil_train)

RMSE(pred, crude_oil_train$closing_price)
```

```
## [1] 2.572239
```

```
predicted_price_algorithm_09 <- predict(ranger_time, newdata = crude_oil_test)

RMSE09 <- RMSE(predicted_price_algorithm_09, crude_oil_test$closing_price)

RMSE09
```

```
## [1] 3.772397
```

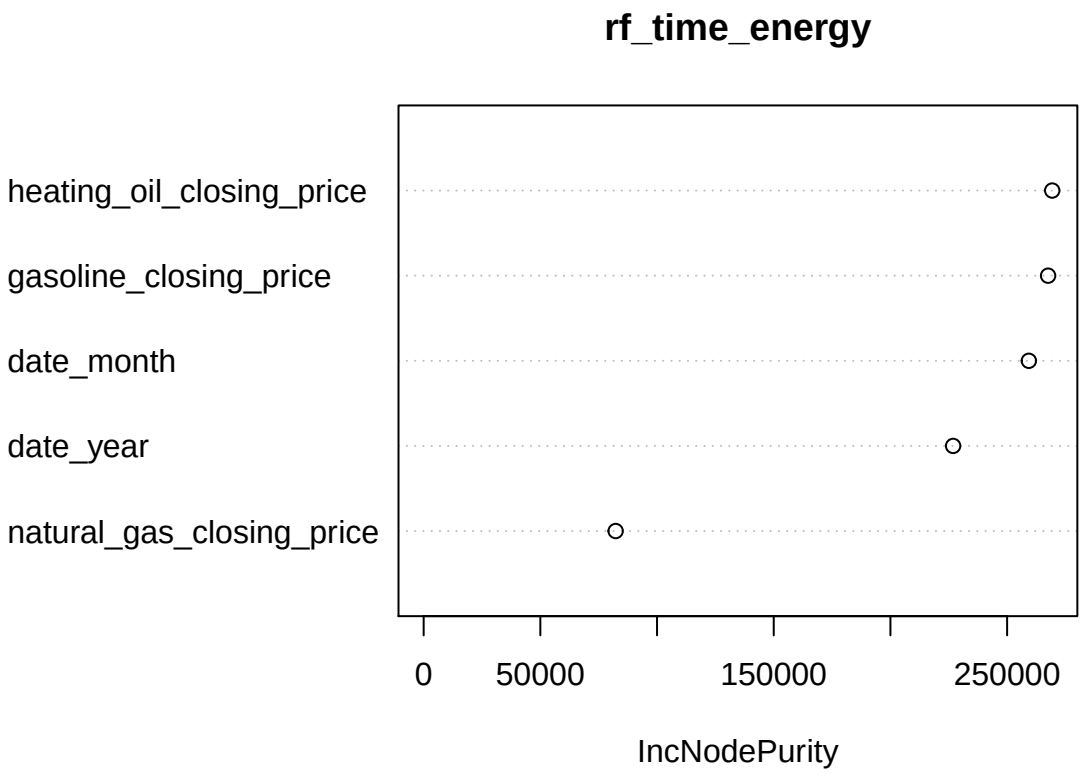
```
### Each of the next three algorithms considers crude oil in comparison to other commodities from the e
### The available predictors are the key components of time and the closing prices of natural gas, heat

#### ALGORITHM 10 (RANDOM FOREST - TIME AND ENERGY)
##### This algorithm predicts crude oil closing prices based on a Random Forest of time components and
####### The first step is to look at a random forest incorporating year, month, and closing prices of n

nodesize <- seq(1, 5, 1)

rmses <- sapply(nodesize, function(n){
  rf_time_energy = randomForest(closing_price ~
                                  date_year +
                                  date_month +
                                  natural_gas_closing_price +
                                  heating_oil_closing_price +
                                  gasoline_closing_price,
                                data = crude_oil_train, nodesize = n)
  pred <- predict(rf_time_energy, newdata = crude_oil_train)
  RMSE(pred, crude_oil_train$closing_price)
})

qplot(nodesize, rmses)
```



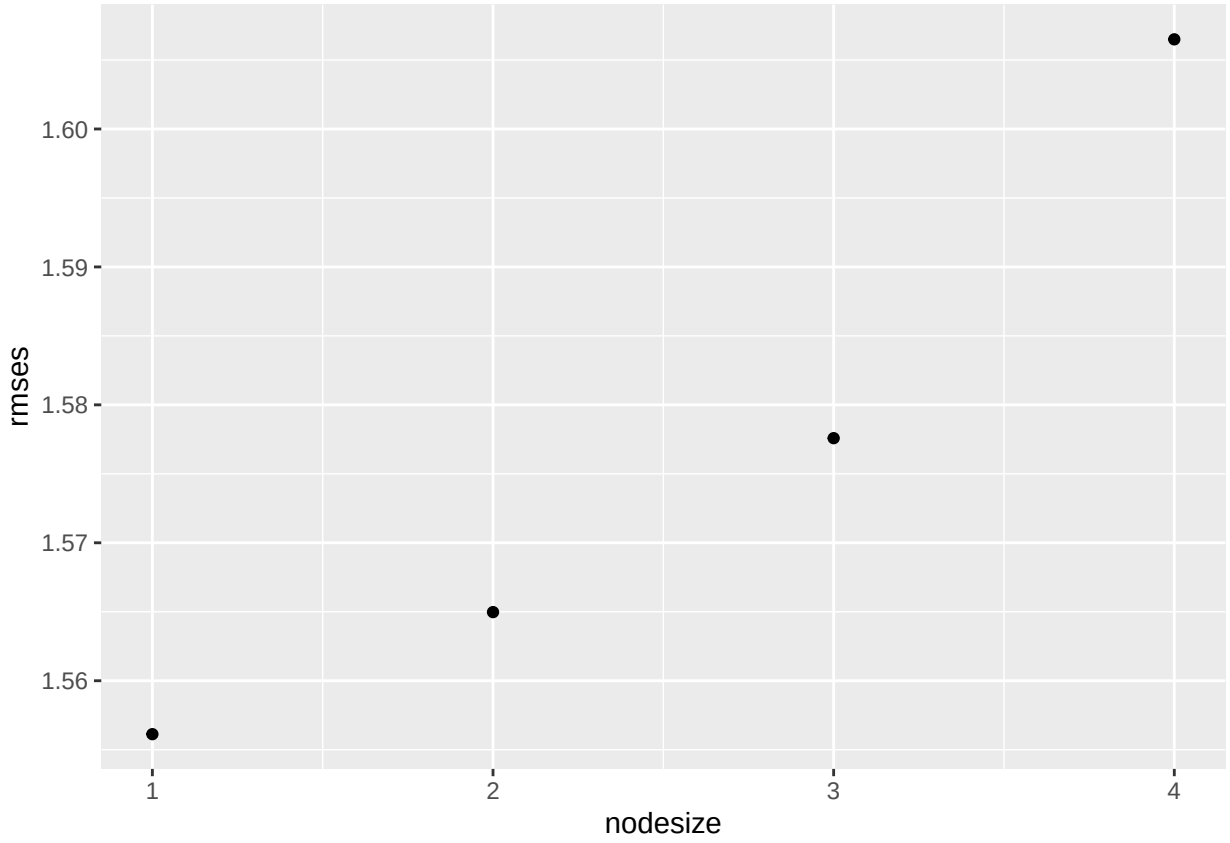

```
nodesize[which.min(rmses)]
```

```
## [1] 2
```

```
rf_time_energy = randomForest(closing_price ~
                                date_year +
                                date_month +
                                natural_gas_closing_price +
                                heating_oil_closing_price +
                                gasoline_closing_price,
                              data = crude_oil_train, nodesize = nodesize[which.min(rmses)])

##### The second is to build a plot to help determine which of those components is/are the most importa

varImpPlot(rf_time_energy)
```




```
####### Based on that plot, Random Forest will be revised to focus on year, month, heating oil, and gas

nodesize <- seq(1, 4, 1)

rmses <- sapply(nodesize, function(n){
  rf_time_energy = randomForest(closing_price ~
                                  date_year +
                                  date_month +
                                  heating_oil_closing_price +
                                  gasoline_closing_price,
                                data = crude_oil_train, nodesize = n)
  pred <- predict(rf_time_energy, newdata = crude_oil_train)
  RMSE(pred, crude_oil_train$closing_price)
})

qplot(nodesize, rmses)
```




```
nodesize[which.min(rmses)]
```

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

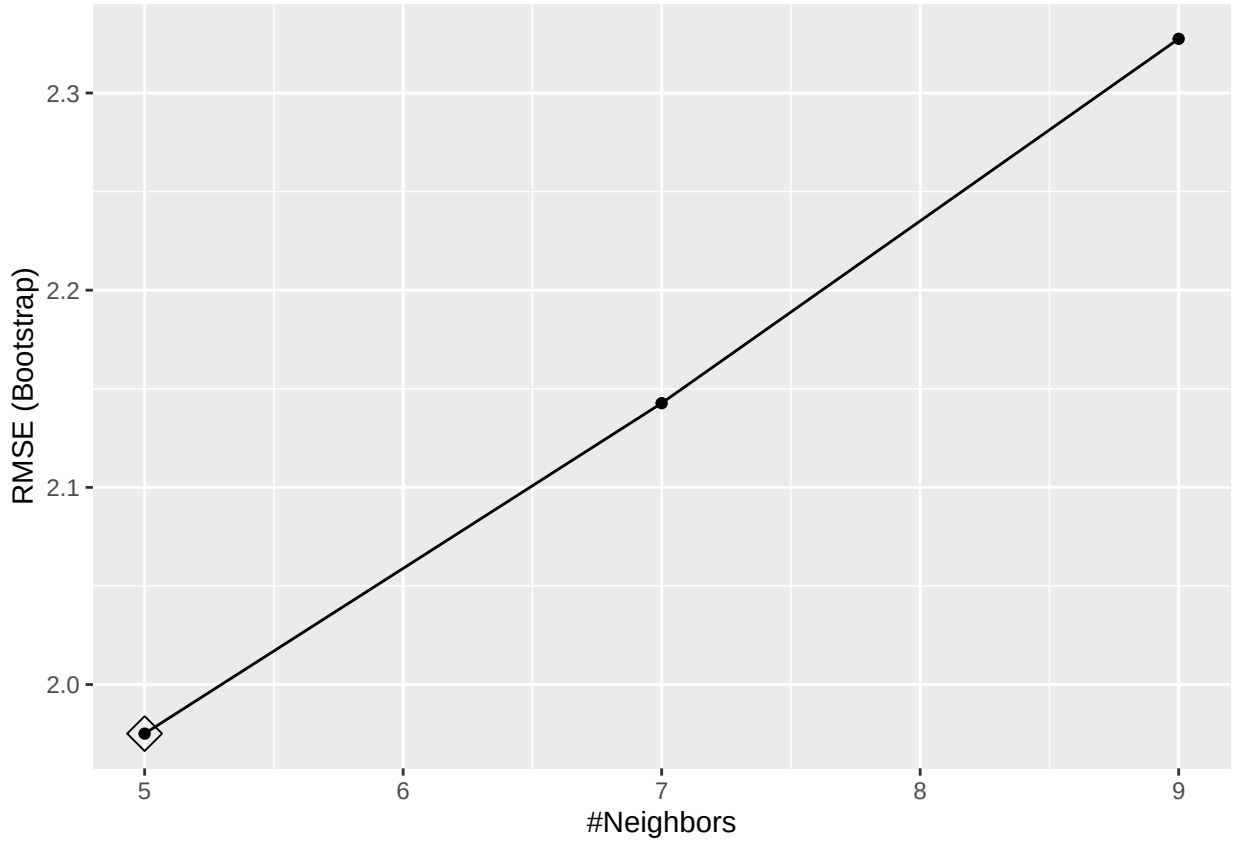
```
rf_time_energy = randomForest(closing_price ~
                                date_year +
                                date_month +
                                heating_oil_closing_price +
                                gasoline_closing_price,
                              data = crude_oil_train, nodesize = nodesize[which.min(rmses)])

pred <- predict(rf_time_energy, newdata = crude_oil_train)

RMSE(pred, crude_oil_train$closing_price)
```

```
## [1] 1.55598
```

```
predicted_price_algorithm_10 <- predict(rf_time_energy, newdata = crude_oil_test)

RMSE10 <- RMSE(predicted_price_algorithm_10, crude_oil_test$closing_price)

RMSE10
```

```
## [1] 3.297452
```

```
#### ALGORITHM 11 (KNN - TIME AND ENERGY)
##### This algorithm predicts crude oil closing prices based on a KNN model of the four key time and en

knn_time_energy <- train(closing_price ~
                           date_year +
                           date_month +
                           heating_oil_closing_price +
                           gasoline_closing_price,
                         method = "knn",
                         data = crude_oil_train)

ggplot(knn_time_energy, highlight = TRUE)
```




```
pred <- predict(knn_time_energy, newdata = crude_oil_train)

RMSE(pred, crude_oil_train$closing_price)
```

```
## [1] 1.337656
```

```
predicted_price_algorithm_11 <- predict(knn_time_energy, newdata = crude_oil_test)

RMSE11 <- RMSE(predicted_price_algorithm_11, crude_oil_test$closing_price)

RMSE11
```

```
## [1] 2.956255
```

```
#### ALGORITHM 12 (RANGER - TIME AND ENERGY)
##### This algorithm predicts crude oil closing prices based on a RANGER model of the key time and ener

ranger_time_energy<- train(closing_price ~
                             date +
                             date_year +
                             heating_oil_closing_price +
                             gasoline_closing_price,
                           method = "ranger",
                           data = crude_oil_train)

pred <- predict(ranger_time_energy, newdata = crude_oil_train)

RMSE(pred, crude_oil_train$closing_price)
```

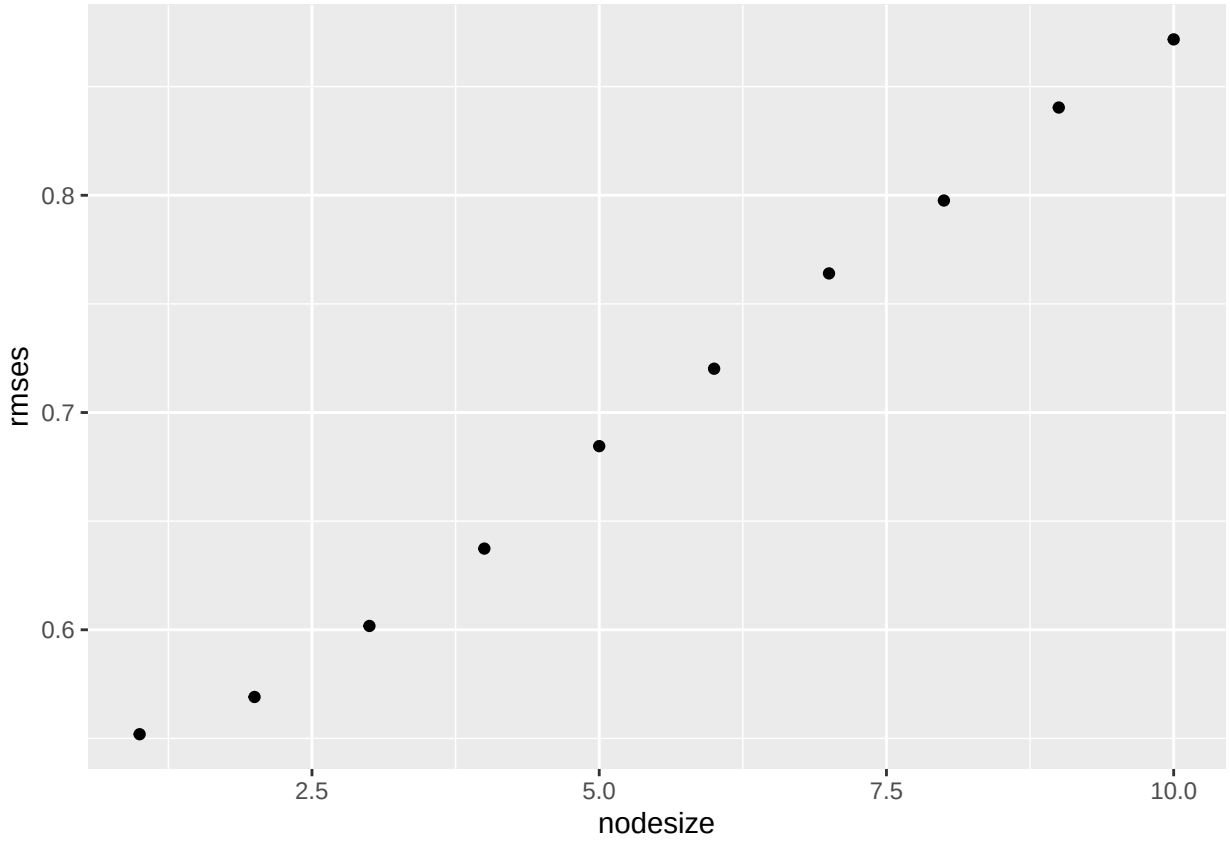
```
## [1] 0.672647
```

```
predicted_price_algorithm_12 <- predict(ranger_time_energy, newdata = crude_oil_test)

RMSE12 <- RMSE(predicted_price_algorithm_12, crude_oil_test$closing_price)

RMSE12
```

```
## [1] 2.963854
```

```
### Each of the final three algorithms considers crude oil in comparison to other commodities from the
### The predictors include not only time but the closing prices of natural gas, heating oil, gasoline,

#### ALGORITHM 13 (RANDOM FOREST - TIME, ENERGY, PRECIOUS METALS, AGRICULTURE)
##### This algorithm predicts crude oil closing prices based on a Random Forest of time, energy, precio
####### The first step is to look at a random forest incorporating date, year, and closing prices of ei

nodesize <- seq(1, 10, 1)

rmses <- sapply(nodesize, function(n){
  rf_time_energy_metals_agriculture = randomForest(closing_price ~
                                                     date_year +
                                                     date_month +
                                                     heating_oil_closing_price +
                                                     gasoline_closing_price +
                                                     gold_closing_price +
                                                     silver_closing_price +
                                                     platinum_closing_price +
                                                     wheat_closing_price +
                                                     rice_closing_price +
                                                     soybeans_closing_price,
                                                   data = crude_oil_train, nodesize = n)
  pred <- predict(rf_time_energy_metals_agriculture, newdata = crude_oil_train)
  RMSE(pred, crude_oil_train$closing_price)
})
```

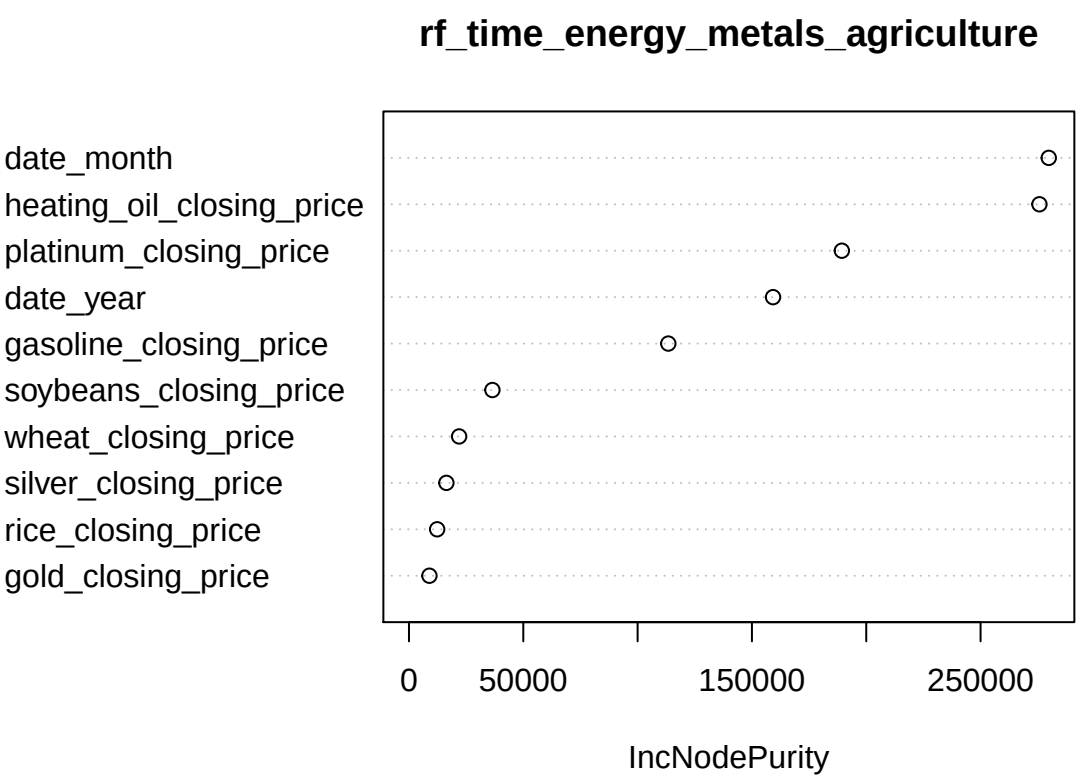
```
qplot(nodesize, rmses)
```



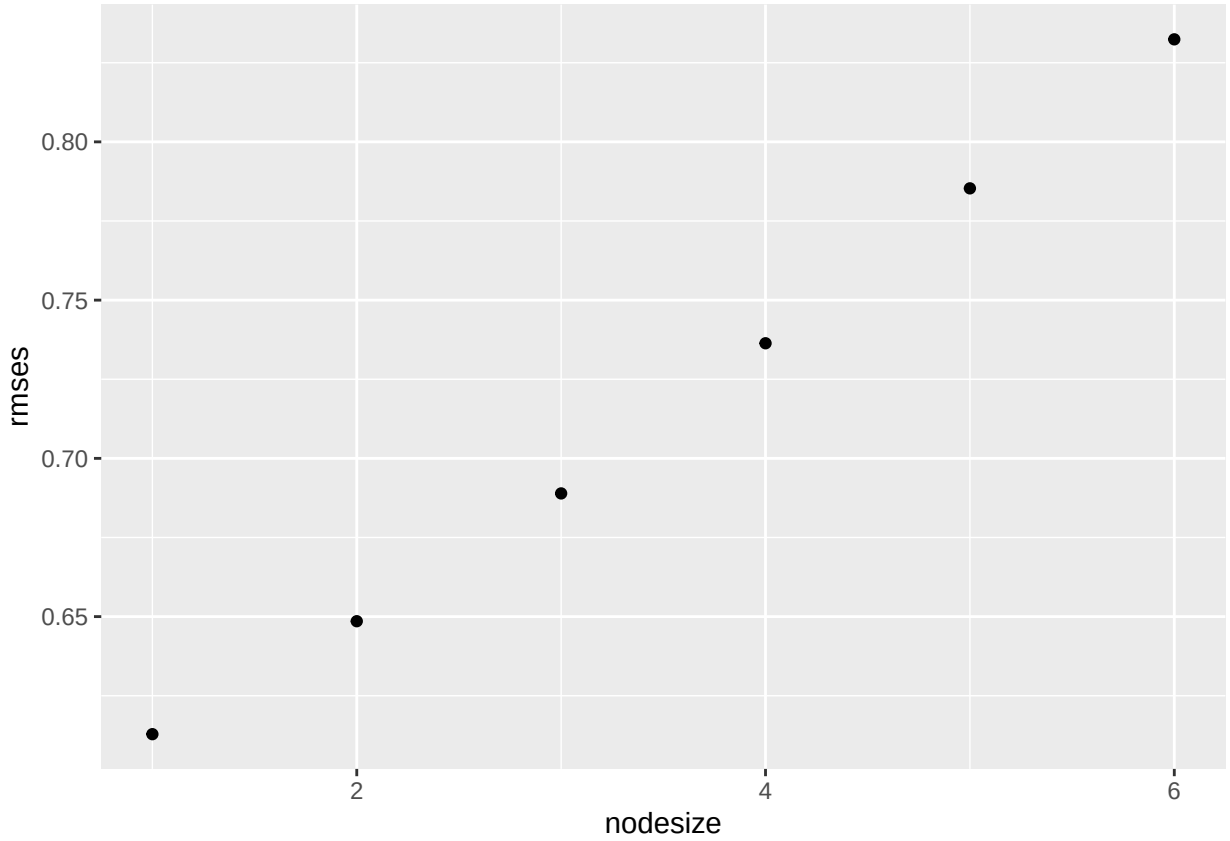

```
nodesize[which.min(rmses)]
```

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

```
rf_time_energy_metals_agriculture = randomForest(closing_price ~
                                                   date_year +
                                                   date_month +
                                                   heating_oil_closing_price +
                                                   gasoline_closing_price +
                                                   gold_closing_price +
                                                   silver_closing_price +
                                                   platinum_closing_price +
                                                   wheat_closing_price +
                                                   rice_closing_price +
                                                   soybeans_closing_price,
                                                 data = crude_oil_train, nodesize = nodesize[which.min(

##### The second is to build a plot to help determine which of those components is/are the most importa

varImpPlot(rf_time_energy_metals_agriculture)
```




```
####### Based on that plot, Random Forest will be revised to focus on year, month, heating oil, gasolin

nodesize <- seq(1, 6, 1)

rmses <- sapply(nodesize, function(n){
  rf_time_energy_metals_agriculture = randomForest(closing_price ~
                                                     date_year +
                                                     date_month +
                                                     heating_oil_closing_price +
                                                     gasoline_closing_price +
                                                     platinum_closing_price +
                                                     soybeans_closing_price,
                                                   data = crude_oil_train, nodesize = n)
  pred <- predict(rf_time_energy_metals_agriculture, newdata = crude_oil_train)
  RMSE(pred, crude_oil_train$closing_price)
})

qplot(nodesize, rmses)
```




```
nodesize[which.min(rmses)]
```

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

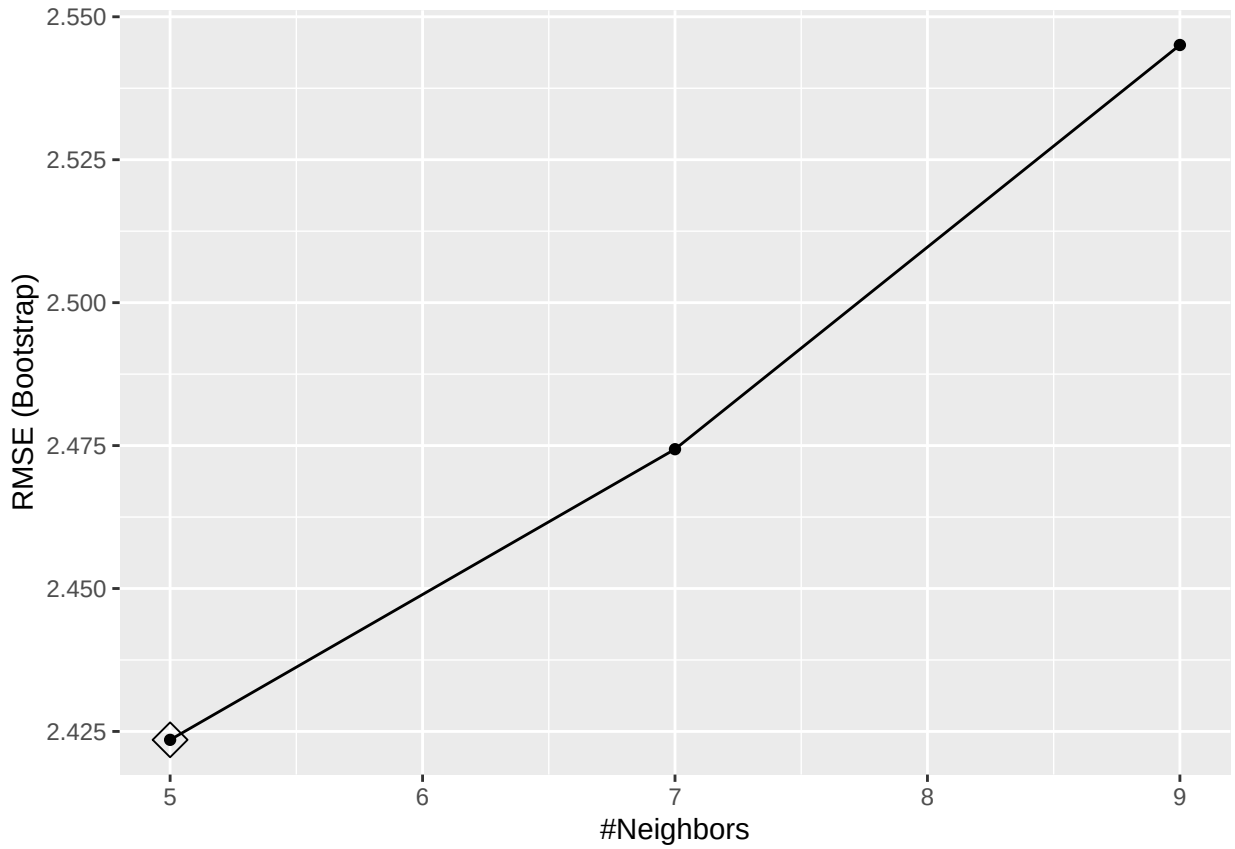
```
rf_time_energy_metals_agriculture = randomForest(closing_price ~
                                                   date_year +
                                                   date_month +
                                                   heating_oil_closing_price +
                                                   gasoline_closing_price +
                                                   platinum_closing_price +
                                                   soybeans_closing_price,
                                                 data = crude_oil_train, nodesize = nodesize[which.min(

pred <- predict(rf_time_energy_metals_agriculture, newdata = crude_oil_train)

RMSE(pred, crude_oil_train$closing_price)
```

```
## [1] 0.6148404
```

```
predicted_price_algorithm_13 <- predict(rf_time_energy_metals_agriculture, newdata = crude_oil_test)

RMSE13 <- RMSE(predicted_price_algorithm_13, crude_oil_test$closing_price)

RMSE13
```

```
## [1] 2.951125
```

```
#### ALGORITHM 14 (KNN - TIME, ENERGY, PRECIOUS METALS, AGRICULTURE)
##### This algorithm predicts crude oil closing prices based on a KNN model incorporating date, heating

knn_time_energy_metals_agriculture <- train(closing_price ~
                                              date_year +
                                              date_month +
                                              heating_oil_closing_price +
                                              gasoline_closing_price +
                                              platinum_closing_price +
                                              soybeans_closing_price,
                                            method = "knn",
                                            data = crude_oil_train)

ggplot(knn_time_energy_metals_agriculture, highlight = TRUE)
```




```
pred <- predict(knn_time_energy_metals_agriculture, newdata = crude_oil_train)

RMSE(pred, crude_oil_train$closing_price)
```

```
## [1] 1.689948
```

```r
predicted_price_algorithm_14 <- predict(knn_time_energy_metals_agriculture, newdata = crude_oil_test)

RMSE14 <- RMSE(predicted_price_algorithm_14, crude_oil_test$closing_price)

RMSE14
```

```
## [1] 3.298945
```

```r
#### ALGORITHM 15 (RANGER - TIME, ENERGY, PRECIOUS METALS, AGRICULTURE)
##### This algorithm predicts crude oil closing prices based on a RANGER model incorporating date, heat

ranger_time_energy_metals_agriculture <- train(closing_price ~
                                                 date_year +
                                                 date_month +
                                                 heating_oil_closing_price +
                                                 gasoline_closing_price +
                                                 platinum_closing_price +
                                                 soybeans_closing_price,
                                               method = "ranger",
                                               data = crude_oil_train)

pred <- predict(ranger_time_energy_metals_agriculture, newdata = crude_oil_train)

RMSE(pred, crude_oil_train$closing_price)
```

```
## [1] 0.8545436
```

```r
predicted_price_algorithm_15 <- predict(ranger_time_energy_metals_agriculture, newdata = crude_oil_test)

RMSE15 <- RMSE(predicted_price_algorithm_15, crude_oil_test$closing_price)

RMSE15
```

```
## [1] 2.930749
```

```r
### SUMMARY OF FINDINGS

ALGORITHM_tab <- c("ALGORITHM 01 - Average Closing Price",
                   "ALGORITHM 02 - Average Closing Price plus Day of the Week Effects",
                   "ALGORITHM 03 - Average Closing Price plus Month of the Year Effects",
                   "ALGORITHM 04 - Average Closing Price plus Quarter of the Year Effects",
                   "ALGORITHM 05 - Average Closing Price plus Year Effects",
                   "ALGORITHM 06 - Average Closing Price Plus Month Effects",
                   "ALGORITHM 07 - Random Forest of Time Components",
                   "ALGORITHM 08 - KNN Model of Time Components",
                   "ALGORITHM 09 - Ranger Model of Time Components",
                   "ALGORITHM 10 - Random Forest of Time and Energy Components",
                   "ALGORITHM 11 - KNN Model of Time and Energy Components",
                   "ALGORITHM 12 - Ranger Model of Time and Energy Components",
                   "ALGORITHM 13 - Random Forest of Time Components and Energy, Precious Metals,and Agri
                   "ALGORITHM 14 - KNN Model of Time Components and Energy, Precious Metals,and Agricul
```

```
                  "ALGORITHM 15 - Ranger Model of Time Components and Energy, Precious Metals,and Agri

RMSE_tab <- c(RMSE01,
              RMSE02,
              RMSE03,
              RMSE04,
              RMSE05,
              RMSE06,
              RMSE07,
              RMSE08,
              RMSE09,
              RMSE10,
              RMSE11,
              RMSE12,
              RMSE13,
              RMSE14,
              RMSE15)

Results <- data.frame(ALGORITHM_tab, RMSE_tab)

Results %>%
  arrange(RMSE_tab) %>%
  print(right = FALSE)
```

```
##    ALGORITHM_tab
## 1  ALGORITHM 15 - Ranger Model of Time Components and Energy, Precious Metals,and Agriculture Commodi
## 2  ALGORITHM 13 - Random Forest of Time Components and Energy, Precious Metals,and Agriculture Commod
## 3  ALGORITHM 11 - KNN Model of Time and Energy Components
## 4  ALGORITHM 12 - Ranger Model of Time and Energy Components
## 5  ALGORITHM 10 - Random Forest of Time and Energy Components
## 6  ALGORITHM 14 - KNN Model of Time Components and Energy, Precious Metals,and Agriculture Commoditi
## 7  ALGORITHM 08 - KNN Model of Time Components
## 8  ALGORITHM 09 - Ranger Model of Time Components
## 9  ALGORITHM 06 - Average Closing Price Plus Month Effects
## 10 ALGORITHM 07 - Random Forest of Time Components
## 11 ALGORITHM 05 - Average Closing Price plus Year Effects
## 12 ALGORITHM 01 - Average Closing Price
## 13 ALGORITHM 02 - Average Closing Price plus Day of the Week Effects
## 14 ALGORITHM 04 - Average Closing Price plus Quarter of the Year Effects
## 15 ALGORITHM 03 - Average Closing Price plus Month of the Year Effects
##    RMSE_tab
## 1   2.930749
## 2   2.951125
## 3   2.956255
## 4   2.963854
## 5   3.297452
## 6   3.298945
## 7   3.768831
## 8   3.772397
## 9   3.800524
## 10  4.716089
## 11  8.268093
## 12 23.512925
```

```
## 13 23.516889
## 14 23.517132
## 15 23.544488
```