Module 2 Assignment, Q4

Student Name

Please read through the Module 2 Assignment 4 document posted in Canvas to get a better understanding of what we are trying to do. This document serves as a submission template for those interested to do Q4 in R. The codes here are suggestions only, and you can feel free to modify them as you wish. Please knit the document to either a html or pdf file and submit that file in Canvas. Remove the `eval = F` for your codes to run. Please ensure the codes are printed in your submission.

Part A

Load Packages and Data

```
pacman::p_load(dplyr, tidyverse, here, lubridate)
```

Using the `here()` function, load the `Q4_data.csv` data saved in the "Data" folder, and call this object `rawdata`.

```
rawdata <- read.csv(here("Data", "Q4_data.csv"))
```

Data Cleaning

If you look at the output of `glimpse(rawdata)`, you will notice that the `Date` variable is stored as character. Use `as.Date()` function to convert this variable into a date data type. Take note of the format (i.e. is the format month/date/year or month-date-year or year-month-date?) You can just overwrite the original `Date` variable. By transforming this variable into a date type, you can now use `{lubridate}`'s `year()` and `month()` functions.

```
glimpse(rawdata)
```

```
## Rows: 1,614
## Columns: 5
## $ Date     <chr> "1/9/2014", "1/16/2014", "1/23/2014", "1/30/2014", "2/6/2014"~
## $ Region   <chr> "Central", "Central", "Central", "Central", "Central", "Centr~
## $ Cash     <dbl> 4.34, 4.45, 4.46, 4.47, 4.60, 4.91, 5.07, 4.99, 5.36, 4.98, 4~
## $ Contract <chr> "Dec", "Dec", "Dec", "Dec", "Dec", "Dec", "Dec", "Dec", "Dec"~
## $ Futures  <dbl> 4.41, 4.52, 4.49, 4.50, 4.58, 4.56, 4.69, 4.61, 4.89, 4.84, 4~
```

```
# convert Date to date format
rawdata$Date <- as.Date(rawdata$Date, format = c("%m/%d/%Y"))

is.Date(rawdata$Date) #you should get true
```

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

Create a new dataframe called `futuresdata` which creates the following variables using data from `rawdata`

- `Year` which is the year of the observation (hint: use the `year()` function)
- `Month` which is the month of the observation (hint: use the `month()` function)
- `Trend` which takes on the value of 0 for January, 1 for February, . . . , 5 for June
- 6 year dummies called `d_2015`, `d_2016`, . . . , `d_2020`
- 2 region dummies called `d_north` and `d_east`
- 2 contract dummies called `d_sept` and `d_dec`
- `Basis` which is the cash price minus the futures price

```
futuresdata <- rawdata %>%
  mutate(year = year(Date),
         month = month(Date),
         trend = month - 1,
         d_2015 = ifelse(year == 2015, 1, 0),
         d_2016 = ifelse(year == 2016, 1, 0),
         d_2017 = ifelse(year == 2017, 1, 0),
         d_2018 = ifelse(year == 2018, 1, 0),
         d_2019 = ifelse(year == 2019, 1, 0),
         d_2020 = ifelse(year == 2020, 1, 0),
         d_north = ifelse(Region == "North", 1, 0),
         d_east = ifelse(Region == "East", 1, 0),
         d_sept = ifelse(Contract == "Sep", 1, 0),
         d_dec = ifelse(Contract == "Dec", 1, 0),
         basis = Cash - Futures)
```

The first four rows of your `futuresdata` dataframe should look like the screenshot provided in the word document. *Hint: use the* ***`head(futuresdata, 4)`*** *command.*

Now estimate a model with `basis` as the dependent variable. The explanatory variables are the `trend` and nine dummies.

```
basis_reg <- lm(basis ~ d_2015 + d_2016 + d_2017 + d_2018 + d_2019 + d_2020 + d_sept + d_dec +
                  d_north + d_east + trend, data = futuresdata)
summary(basis_reg)
```

```
##
## Call:
## lm(formula = basis ~ d_2015 + d_2016 + d_2017 + d_2018 + d_2019 +
##     d_2020 + d_sept + d_dec + d_north + d_east + trend, data = futuresdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.53657 -0.08877  0.00200  0.08690  0.68034
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.067912   0.014589   4.655 3.51e-06 ***
## d_2015      -0.245004   0.015073 -16.254  < 2e-16 ***
## d_2016      -0.338602   0.015073 -22.464  < 2e-16 ***
## d_2017      -0.474692   0.015073 -31.492  < 2e-16 ***
```

```
## d_2018      -0.294629   0.015120 -19.486  < 2e-16 ***
## d_2019      -0.087634   0.015072  -5.814 7.34e-09 ***
## d_2020      -0.136478   0.015073  -9.054  < 2e-16 ***
## d_sept      -0.045943   0.009700  -4.737 2.37e-06 ***
## d_dec       -0.115562   0.009700 -11.914  < 2e-16 ***
## d_north      0.218770   0.009655  22.658  < 2e-16 ***
## d_east       0.113517   0.009741  11.653  < 2e-16 ***
## trend        0.019221   0.002303   8.346  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1591 on 1602 degrees of freedom
## Multiple R-squared:  0.571,  Adjusted R-squared:  0.5681
## F-statistic: 193.8 on 11 and 1602 DF,  p-value: < 2.2e-16
```

To construct a similar looking pivot table, you can use the `filter()` function to only look at data in 2020, `group_by()` function to group observations by `month` and `Contract`, `summarise()` function to calculate the average, and `pivot_wider()` function to convert the table into a long format. You can use the `select()` function to rearrange your columns. The end result will match the screenshot in the document, without the Grand Total column. Call this object `pivot`. *Note: You may get a message that says* ***`summarise has grouped output by month...`*** *and that is fine. If you keep the* ***`message = F`*** *option you will not see this message in your knitted document.*

```
pivot <- futuresdata %>%
  filter(year == 2020) %>%
  group_by(month, Contract) %>%
  summarise(avg_futures = mean(Futures)) %>%
  pivot_wider(names_from = Contract,
              values_from = avg_futures) %>%
  select(month, Jul, Sep, Dec)

pivot
```

```
## # A tibble: 6 x 4
## # Groups:   month [6]
##   month   Jul   Sep   Dec
##   <dbl> <dbl> <dbl> <dbl>
## 1     1  3.96  3.96  3.98
## 2     2  3.84  3.82  3.87
## 3     3  3.64  3.66  3.72
## 4     4  3.30  3.34  3.44
## 5     5  3.20  3.25  3.35
## 6     6  3.27  3.31  3.39
```

Part B

Please read the overview and instructions in the document carefully.

1. The value you should use for $\bar{F}_{July}$ is the average weekly value of the July contract in January of 2020. You can access this value from your `pivot` object. Call this value `july_futures`. See code tips sheet for help.

2. The value of β_P, which is the slope of the expected spot price schedule, is the estimated coefficient on the `Trend` variable you estimated from Part A. Call this value `slope`.
3. The value of $\bar{B}_{Jan}^{Texas}$, which is the sum of the estimated intercept and the estimated coefficient on the 2020 dummy variable. This value can be interpreted as the expected corn basis in Central Texas in January of 2020. Call this value `jan_basis_texas`.

```
july_futures <- pivot[[1,2]]
july_futures
```

```
## [1] 3.965
```

```
slope <- basis_reg$coefficients['trend']
slope
```

```
##      trend
## 0.01922148
```

```
jan_basis_texas <- basis_reg$coefficients['(Intercept)'] + basis_reg$coefficients['d_2020']
jan_basis_texas
```

```
## (Intercept)
## -0.06856647
```

Given these values, you can now build the dataframe with four columns (`month`, `futures`, `texas_spot`, and `chicago_spot`) to create the graph.

1. Create a vector called `months` that contains values "Jan 1, Feb 1, ..., July 1". You can use the `c()` and `as.Date` function. This vector should be recognized as dates for the graph to work correctly.
2. Create a vector called `futures` that contains the value of $\bar{F}_{July}$. It should be obvious that the value for $\bar{F}_{July}$ repeats 7 times (number of elements of the `month` vector). You can use the `rep()` function (see code tips sheet).
3. Create a vector called `texas_spot` Since you can calculate $\bar{P}_{Jan}^{Chicago}$ and you have the `slope` (increment the values increase by) from earlier steps, you can use the `seq()` function (see code tips sheet) to create the `texas_spot` vector.
4. Create a vector called `chicago_spot`. You can use the same approach as above.
5. Given the values you calculated, you can compute for `t*`.
6. Use `data.frame()` to combine the 4 vectors you just created and transform it to a dataframe called `df_graph`.

```
months <- as.Date(c("01-01-2020", "02-01-2020", "03-01-2020", "04-01-2020",
                    "05-01-2020", "06-01-2020", "07-01-2020"), format = c("%m-%d-%Y"))

futures <- rep(july_futures, 7)

jan_spot_texas <- july_futures + jan_basis_texas
texas_spot <- seq(jan_spot_texas, by = slope, length = 7)

jan_spot_chicago <- jan_spot_texas - (texas_spot[7] - july_futures)
chicago_spot <- seq(jan_spot_chicago, july_futures, by = slope)

t_star <- (july_futures - jan_spot_texas)/slope
t_star
```

```
## (Intercept)
##    3.567179
```

```
df_graph <- data.frame(months, futures, texas_spot, chicago_spot)
df_graph
```

```
##       months futures texas_spot chicago_spot
## 1 2020-01-01   3.965   3.896434     3.849671
## 2 2020-02-01   3.965   3.915655     3.868893
## 3 2020-03-01   3.965   3.934876     3.888114
## 4 2020-04-01   3.965   3.954098     3.907336
## 5 2020-05-01   3.965   3.973319     3.926557
## 6 2020-06-01   3.965   3.992541     3.945779
## 7 2020-07-01   3.965   4.011762     3.965000
```

Now use `ggplot() + geom_line()` to recreate the graph. *Hint: (1) Since you will plot 3 different lines, you can use* ***`geom_line()`*** *three times. (2) If your* ***`months`*** *variable is in the date format, you can use 'scale_x_date(date_lables = "%b %d", date_breaks = "1 month") to show all 7 date mark ticks on your Y axis and the labels will be in Jan 1, Feb 1, . . . , Jul 1 format. See here for info.*

```
ggplot(df_graph, aes(x = months)) +
  geom_line(aes(y = futures), color = "blue") +
  geom_line(aes(y = texas_spot), color = "orange") +
  geom_line(aes(y = chicago_spot), color = "darkgray") +
  labs(x = "Date", y = "Price", title = "Expected Corn July Futures and Spot") +
  scale_x_date(date_labels = "%b %d", date_breaks  ="1 month")
```