Q4 code tips sheet

Accessing values in a regression

Here are some codes you can use to access the value of the coefficients after running a regression. We first load the `iris` (a type of flower) dataset and run a regression to determine the factors that affect the length of an iris petal.

```
# load packages
pacman::p_load(dplyr, tidyverse)

# load data
data(iris)

# look at what the data is like
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
# run a linear model
model1 <- lm(Petal.Width ~ Sepal.Length + Sepal.Width + Petal.Length, data = iris)
summary(model1)
```

```
##
## Call:
## lm(formula = Petal.Width ~ Sepal.Length + Sepal.Width + Petal.Length,
##     data = iris)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.60959 -0.10134 -0.01089  0.09825  0.60685
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.24031    0.17837  -1.347     0.18
## Sepal.Length -0.20727    0.04751  -4.363 2.41e-05 ***
## Sepal.Width   0.22283    0.04894   4.553 1.10e-05 ***
## Petal.Length  0.52408    0.02449  21.399  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.192 on 146 degrees of freedom
## Multiple R-squared:  0.9379, Adjusted R-squared:  0.9366
## F-statistic: 734.4 on 3 and 146 DF,  p-value: < 2.2e-16
```

```
# get the intercept
model1$coefficients[1]
```

```
## (Intercept)
##  -0.2403074
```

```
# get coefficient of petal.length
model1$coefficients['Petal.Length']
```

```
## Petal.Length
##    0.5240831
```

```
# to get the coefficients of sepal.length and sepal.width (second and third variables in the reg)
model1$coefficients[2:3]
```

```
## Sepal.Length  Sepal.Width
##   -0.2072661    0.2228285
```

Pivot Tables in R

You can use `group_by()` and `summarise()` functions to replicate Pivot Tables in R. The `iris` dataset contains 3 species (`table(iris$Species)`) and we want to see the average petal length and width of each species.

```
petal_pivot <- iris %>%
  group_by(Species) %>%
  summarise(avg_width = mean(Petal.Width),
            avg_length = mean(Petal.Length))

petal_pivot
```

```
## # A tibble: 3 x 3
##   Species    avg_width avg_length
##   <fct>          <dbl>      <dbl>
## 1 setosa         0.246       1.46
## 2 versicolor     1.33        4.26
## 3 virginica      2.03        5.55
```

The `petal_pivot` object is a tibble, which is a special type of a dataframe (if you type `class(petal_pivot)`, it will say `tbl_df`). To access values in a tibble, you will have to use double brackets instead of single brackets like you would with a data frame. Alternatively, you can call use the variable name with the `$` operator to look at the `avg_length` column and use single brackets to extract the value for the setosa row.

```
petal_pivot[[1, 3]]
```

```
## [1] 1.462
```

```
petal_pivot$avg_length[petal_pivot$Species == "setosa"]
```

```
## [1] 1.462
```

Creating vectors using `rep()` and `seq()`

To replicate elements of vectors, such as your July futures price, you can use the `rep()` function (documentation here). The syntax is `rep(value, number_of_times)`. The code to create a vector that contains the value 2 ten times would be

```
repeating_vector <- rep(2, 10)
repeating_vector
```

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

To generate a sequence of numbers, such as in the Chicago and Texas spot prices, you may want to use the `seq()` function (see documentation here). The syntax is `seq(start_value, end_value, by = increment_value)`. The code to create a vector that starts at 1 and increases by the value of 3 until it reaches the value 20 is

```
sequence <- seq(1, 20, by = 3)
sequence
```

```
## [1]  1  4  7 10 13 16 19
```

You can also use the `seq()` function if you only know the starting value, the increment, and the desired length of the sequence. The code to create a vector that starts at 1 and increases by the value of 3 until the vector reaches the length of 7 is

```
sequence2 <- seq(1, by = 3, length = 7)
sequence2
```

```
## [1]  1  4  7 10 13 16 19
```