Final Project

Trong-Thuc Trang

12/6/2020

Generate the dataframe turnip_price.

```
turnip_price_temp <- read_excel("Turnip price record 04262020 -
07182020.xlsx")

## New names:
## * `` -> ...1

colnames(turnip_price_temp) <- c("Date_Time","Buying Price","Selling price
(morning)","Selling price (afternoon)")
turnip_price <- as.data.frame(turnip_price_temp[, 1:3])
colnames(turnip_price) <- c("Date_Time", "Buying_Price", "Selling_Price")
i <- 1
j <- i
while (i <= nrow(turnip_price)) {
  if ((is.na(turnip_price[i, 2]) == TRUE & is.na(turnip_price[i, 3]) != TRUE)
| (mday(turnip_price[i, 1]) == 1 && month(turnip_price[i, 1]) == 5)) {
    # Add the selling price in the afternoon and change the time to 12:00 pm
    turnip_price[(i+1):(nrow(turnip_price)+1), ] <-
turnip_price[i:nrow(turnip_price), ]
    turnip_price[i+1, 3] <- turnip_price_temp[j, 4]
    hour(turnip_price[i+1, 1]) <- 12
    # Skip the current and the new rows
    i <- i + 2
  }
  else
    # Skip the current row
    i <- i + 1
  j <- j + 1
}
print(turnip_price)

##                Date_Time Buying_Price Selling_Price
## 1   2020-04-26 00:00:00          110            NA
## 2   2020-04-27 00:00:00           NA            96
## 3   2020-04-27 12:00:00           NA            92
## 4   2020-04-28 00:00:00           NA           138
## 5   2020-04-28 12:00:00           NA           220
## 6   2020-04-29 00:00:00           NA           385
## 7   2020-04-29 12:00:00           NA           201
## 8   2020-04-30 00:00:00           NA           123
## 9   2020-04-30 12:00:00           NA            90
## 10  2020-05-01 00:00:00            NA            NA
## 11  2020-05-01 12:00:00            NA            NA
## 12  2020-05-02 00:00:00            NA            58
## 13  2020-05-02 12:00:00            NA            52
## 14  2020-05-03 00:00:00            98            NA
## 15  2020-05-04 00:00:00            NA            84
## 16  2020-05-04 12:00:00            NA            80
## 17  2020-05-05 00:00:00            NA            76
## 18  2020-05-05 12:00:00            NA            72
## 19  2020-05-06 00:00:00            NA            68
## 20  2020-05-06 12:00:00            NA            63
## 21  2020-05-07 00:00:00            NA            60
## 22  2020-05-07 12:00:00            NA            55
## 23  2020-05-08 00:00:00            NA            51
## 24  2020-05-08 12:00:00            NA            48
## 25  2020-05-09 00:00:00            NA            44
## 26  2020-05-09 12:00:00            NA            40
## 27  2020-05-10 00:00:00            94            NA
## 28  2020-05-11 00:00:00            NA           108
## 29  2020-05-11 12:00:00            NA           121
## 30  2020-05-12 00:00:00            NA           131
## 31  2020-05-12 12:00:00            NA           125
## 32  2020-05-13 00:00:00            NA           112
## 33  2020-05-13 12:00:00            NA            98
## 34  2020-05-14 00:00:00            NA            59
## 35  2020-05-14 12:00:00            NA            53
## 36  2020-05-15 00:00:00            NA           117
## 37  2020-05-15 12:00:00            NA            69
## 38  2020-05-16 00:00:00            NA            63
## 39  2020-05-16 12:00:00            NA            56
## 40  2020-05-17 00:00:00           104            NA
## 41  2020-05-18 00:00:00            NA            89
## 42  2020-05-18 12:00:00            NA            84
## 43  2020-05-19 00:00:00            NA            79
## 44  2020-05-19 12:00:00            NA            76
## 45  2020-05-20 00:00:00            NA            72
## 46  2020-05-20 12:00:00            NA            68
## 47  2020-05-21 00:00:00            NA            65
## 48  2020-05-21 12:00:00            NA            60
## 49  2020-05-22 00:00:00            NA            56
## 50  2020-05-22 12:00:00            NA            51
## 51  2020-05-23 00:00:00            NA            47
## 52  2020-05-23 12:00:00            NA            42
## 53  2020-05-24 00:00:00            90            NA
## 54  2020-05-25 00:00:00            NA            79
## 55  2020-05-25 12:00:00            NA            75
## 56  2020-05-26 00:00:00            NA            92
## 57  2020-05-26 12:00:00            NA           156
## 58  2020-05-27 00:00:00            NA           534
## 59  2020-05-27 12:00:00            NA           170
## 60  2020-05-28 00:00:00          NA            97
## 61  2020-05-28 12:00:00          NA            77
## 62  2020-05-29 00:00:00          NA            47
## 63  2020-05-29 12:00:00          NA            80
## 64  2020-05-30 00:00:00          NA            57
## 65  2020-05-30 12:00:00          NA            55
## 66  2020-05-31 00:00:00         103            NA
## 67  2020-06-01 00:00:00          NA           105
## 68  2020-06-01 12:00:00          NA           139
## 69  2020-06-02 00:00:00          NA           113
## 70  2020-06-02 12:00:00          NA            63
## 71  2020-06-03 00:00:00          NA            53
## 72  2020-06-03 12:00:00          NA            44
## 73  2020-06-04 00:00:00          NA           135
## 74  2020-06-04 12:00:00          NA           126
## 75  2020-06-05 00:00:00          NA            95
## 76  2020-06-05 12:00:00          NA            82
## 77  2020-06-06 00:00:00          NA            85
## 78  2020-06-06 12:00:00          NA           140
## 79  2020-06-07 00:00:00         105            NA
## 80  2020-06-08 00:00:00          NA            57
## 81  2020-06-08 12:00:00          NA            53
## 82  2020-06-09 00:00:00          NA           124
## 83  2020-06-09 12:00:00          NA           109
## 84  2020-06-10 00:00:00          NA           156
## 85  2020-06-10 12:00:00          NA           167
## 86  2020-06-11 00:00:00          NA           154
## 87  2020-06-11 12:00:00          NA            77
## 88  2020-06-12 00:00:00          NA            73
## 89  2020-06-12 12:00:00          NA            69
## 90  2020-06-13 00:00:00          NA            65
## 91  2020-06-13 12:00:00          NA            61
## 92  2020-06-14 00:00:00          91            NA
## 93  2020-06-15 00:00:00          NA            79
## 94  2020-06-15 12:00:00          NA            76
## 95  2020-06-16 00:00:00          NA            73
## 96  2020-06-16 12:00:00          NA            69
## 97  2020-06-17 00:00:00          NA            66
## 98  2020-06-17 12:00:00          NA            62
## 99  2020-06-18 00:00:00          NA            58
## 100 2020-06-18 12:00:00          NA            54
## 101 2020-06-19 00:00:00          NA            50
## 102 2020-06-19 12:00:00          NA            47
## 103 2020-06-20 00:00:00          NA            44
## 104 2020-06-20 12:00:00          NA            40
## 105 2020-06-21 00:00:00          94            NA
## 106 2020-06-22 00:00:00          NA            81
## 107 2020-06-22 12:00:00          NA            77
## 108 2020-06-23 00:00:00          NA           123
## 109 2020-06-23 12:00:00          NA           144
## 110 2020-06-24 00:00:00          NA          523
## 111 2020-06-24 12:00:00          NA          183
## 112 2020-06-25 00:00:00          NA          100
## 113 2020-06-25 12:00:00          NA           57
## 114 2020-06-26 00:00:00          NA           64
## 115 2020-06-26 12:00:00          NA           71
## 116 2020-06-27 00:00:00          NA           65
## 117 2020-06-27 12:00:00          NA           64
## 118 2020-06-28 00:00:00         103           NA
## 119 2020-06-29 00:00:00          NA           86
## 120 2020-06-29 12:00:00          NA          128
## 121 2020-06-30 00:00:00          NA          111
## 122 2020-06-30 12:00:00          NA          147
## 123 2020-07-01 00:00:00          NA          167
## 124 2020-07-01 12:00:00          NA          159
## 125 2020-07-02 00:00:00          NA           63
## 126 2020-07-02 12:00:00          NA           58
## 127 2020-07-03 00:00:00          NA           53
## 128 2020-07-03 12:00:00          NA           49
## 129 2020-07-04 00:00:00          NA           45
## 130 2020-07-04 12:00:00          NA           40
## 131 2020-07-05 00:00:00         103           NA
## 132 2020-07-06 00:00:00          NA          121
## 133 2020-07-06 12:00:00          NA          109
## 134 2020-07-07 00:00:00          NA          143
## 135 2020-07-07 12:00:00          NA          143
## 136 2020-07-08 00:00:00          NA           69
## 137 2020-07-08 12:00:00          NA           60
## 138 2020-07-09 00:00:00          NA          136
## 139 2020-07-09 12:00:00          NA          111
## 140 2020-07-10 00:00:00          NA           72
## 141 2020-07-10 12:00:00          NA           71
## 142 2020-07-11 00:00:00          NA           62
## 143 2020-07-11 12:00:00          NA          113
## 144 2020-07-12 00:00:00          92           NA
## 145 2020-07-13 00:00:00          NA           82
## 146 2020-07-13 12:00:00          NA           77
## 147 2020-07-14 00:00:00          NA           91
## 148 2020-07-14 12:00:00          NA          183
## 149 2020-07-15 00:00:00          NA          405
## 150 2020-07-15 12:00:00          NA          148
## 151 2020-07-16 00:00:00          NA          112
## 152 2020-07-16 12:00:00          NA           57
## 153 2020-07-17 00:00:00          NA           65
## 154 2020-07-17 12:00:00          NA           71
## 155 2020-07-18 00:00:00          NA           82
## 156 2020-07-18 12:00:00          NA           59
```

1. Visualization of `turnip_price`.

(a) Method: Here I use `ggplot` to show the scatter plot of the data points as well as lines connecting data points of the same variable.

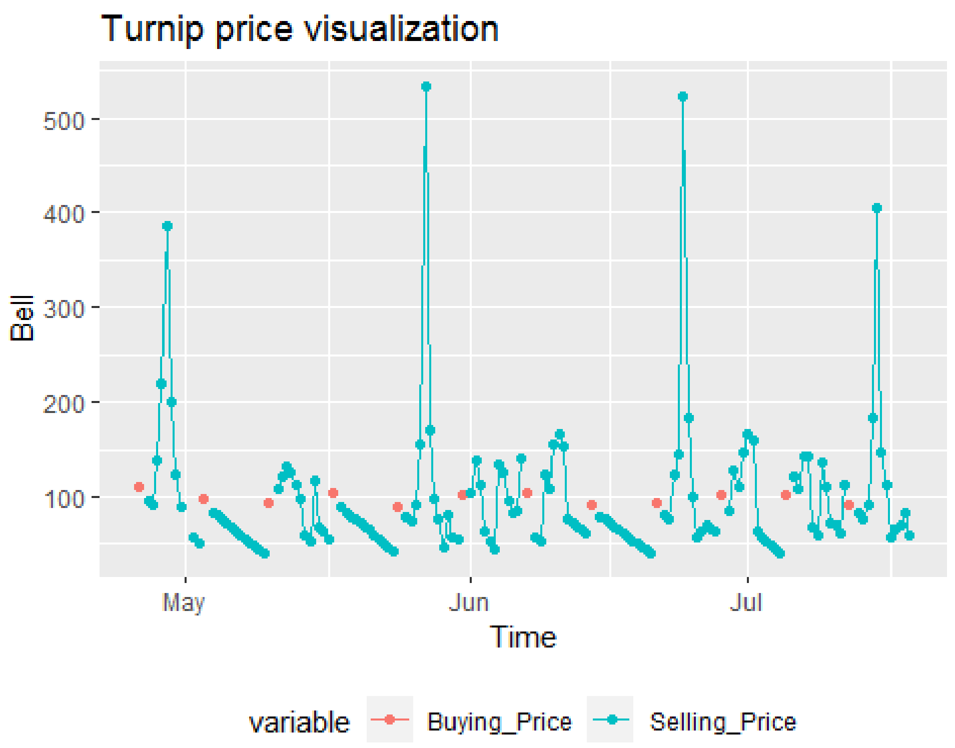
(b) Reason of using this method: I wanted to see the patterns of the variation of the buying and the selling prices.

(c) R code:

```
turnip_price_plot <- melt(turnip_price,  id.vars = 'Date_Time')
ggplot(turnip_price_plot, aes(x = Date_Time, y = value, colour = variable)) +
  geom_point(position = position_dodge(width = .3)) +
  geom_line(aes(colour = variable)) +
  labs(title = "Turnip price visualization", x = "Time", y = "Bell") +
  theme(legend.position="bottom")
```

```
## Warning: Removed 158 rows containing missing values (geom_point).

## Warning: Removed 13 row(s) containing missing values (geom_path).
```



(d) The result generated by the above R code: in the report file.

(e) Summary of the visualization:

- If the price decreases or increases, it will decrease or increase twice a day.
- There were weeks in which the price only decreased.

- If we look at the highest peaks, we will see that, usually, there will be 3 large increases prior to each of the highest peaks.

(f) Explanation of the visualization: Because the players can only buy on Sundays and only sell every other day, the data points representing the `Buying price` is not connected. Also, in the first week of the data, the connecting line representing `Selling price` has a gap right on the date 5/1 because the Stalk market was closed as well as gaps every Sunday because of the same reason mentioned earlier.

2. Mean and range of buying price:

(a) Method: Here I use `mean` and `range` functions with all NA observations of `Buying Price` are removed.

(b) Reason of using this method: The reason why we use these functions is quite natural and we need to set `na.rm = TRUE` for the sake of our computation because we only want the observations with numeric values to be computed.

(c) R code:

```r
print(mean(turnip_price$Buying_Price, na.rm = TRUE))

## [1] 98.91667

print(range(turnip_price$Buying_Price, na.rm = TRUE))

## [1]  90 110
```

(d) The results generated by the above R code: in the report file.

(e) Summary of the results: The `range` of the buying price does not seem to vary much with the following standard deviation.

```r
print(sd(turnip_price$Buying_Price, na.rm = TRUE))

## [1] 6.570711
```

(f) Explanation of the results: Maybe the game has a fixed range of buying price and then every Sunday the buying price will be randomized between this range. The reason is probably because the designers wanted everyone to be able to buy white turnip with a reasonable and stable price.

3. Mean and range of buying price:

(a) Method: Just like previous question, I use `mean` and `range` functions with all NA observations of `Selling Price` are removed.

(b) Reason of using this method: The reason is similar to the previous question, which is quite natural as the names of the functions suggest. Besides, we need to set `na.rm = TRUE` for the sake of our computation because we only want the observations with numeric values to be computed.

(c) R code:

```r
print(mean(turnip_price$Selling_Price, na.rm = TRUE))

## [1] 98.44366

print(range(turnip_price$Selling_Price, na.rm = TRUE))

## [1]  40 534
```

(d) The results generated by the above R code: in the report file.

(e) Summary of the results: It is interesting that the computed `mean` of the selling price is approximately equal to that of the buying price. Meanwhile, the `range` of the selling price varies drastically from `40` to `534` with the following standard deviation.

```r
print(sd(turnip_price$Selling_Price, na.rm = TRUE))

## [1] 73.42092
```

(f) Explanation of the results: I guess the designer of this game tried to mimic the stock market in an ideal way because as we can see in the previous question the buying price is quite stable, but the selling price varies more drastically. With these settings, any player can probably try to predict the rules of choosing when to sell and buy white turnips to make more profit or at least not to lose so much.

4. First, we set the null hypothesis H_0 = “The turnip selling price in the afternoon is not significantly higher than the turnip selling price in the morning”. Then the alternative hypothesis H_a = “The turnip selling price in the afternoon is significantly higher than the turnip selling price in the morning” which is the hypothesis in question.

(a) Method: We use t-test for paired samples in this case.

(b) Reason of using this method: First, the data we are working on was collected by gathering the selling price of a white turnip twice a day: one in the morning and the other in the afternoon. Besides, we can omit the step of checking if the data is normally distributed since its size is greater than `30`.

(c) R code:

```r
morning_price <- subset(turnip_price, hour(Date_Time) == 0, Selling_Price)
morning_price <- drop_na(morning_price)
afternoon_price <- subset(turnip_price, hour(Date_Time) == 12, Selling_Price)
afternoon_price <- drop_na(afternoon_price)
result_q4 <- t.test(afternoon_price$Selling_Price,
morning_price$Selling_Price, alternative = "greater", paired = TRUE)
print(result_q4)

##
##  Paired t-test
##
## data:  afternoon_price$Selling_Price and morning_price$Selling_Price
## t = -1.9481, df = 70, p-value = 0.9723
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  -31.54621       Inf
## sample estimates:
## mean of the differences
##                     -17
```

(d) The result generated by the above R code: in the report file.

(e) Summary of the result: Since `p-value = 0.9723` which is greater than the significance level `alpha = 0.05`, we cannot reject the null hypothesis H_0. This means we cannot conclude that the turnip selling price in the afternoon is significantly higher than the turnip selling price in the morning.

(f) Explanation of the result: If we look at the data or its visualization in the first question, we can see 2 main reasons that probably cause the above result. First, if the selling price in the afternoon is significantly higher than the selling price in the morning, then the selling price in the next morning will grow very steeply and then drop very rapidly in the afternoon of the same day. Second, if the selling price in the morning is lower as compared to the selling price in the afternoon before that, the the selling price in that afternoon will likely drop. It turns out the morning selling price is the significantly higher one between the two. Evidently, we can run the t-test to verify the new null hypothesis H_0 = "The morning selling price is not significantly higher than the afternoon selling price".

```
result_q4 <- t.test(morning_price$Selling_Price,
afternoon_price$Selling_Price, alternative = "greater", paired = TRUE)
print(result_q4)

##
##  Paired t-test
##
## data:  morning_price$Selling_Price and afternoon_price$Selling_Price
## t = 1.9481, df = 70, p-value = 0.02771
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  2.453795      Inf
## sample estimates:
## mean of the differences
##                      17
```

Since `p-value = 0.02771` which is less than the significance level `alpha = 0.05`, we can reject the null hypothesis H_0. This means we can conclude that the morning selling price of a turnip is significantly higher than its afternoon selling price.

5. Imputation of missing value:

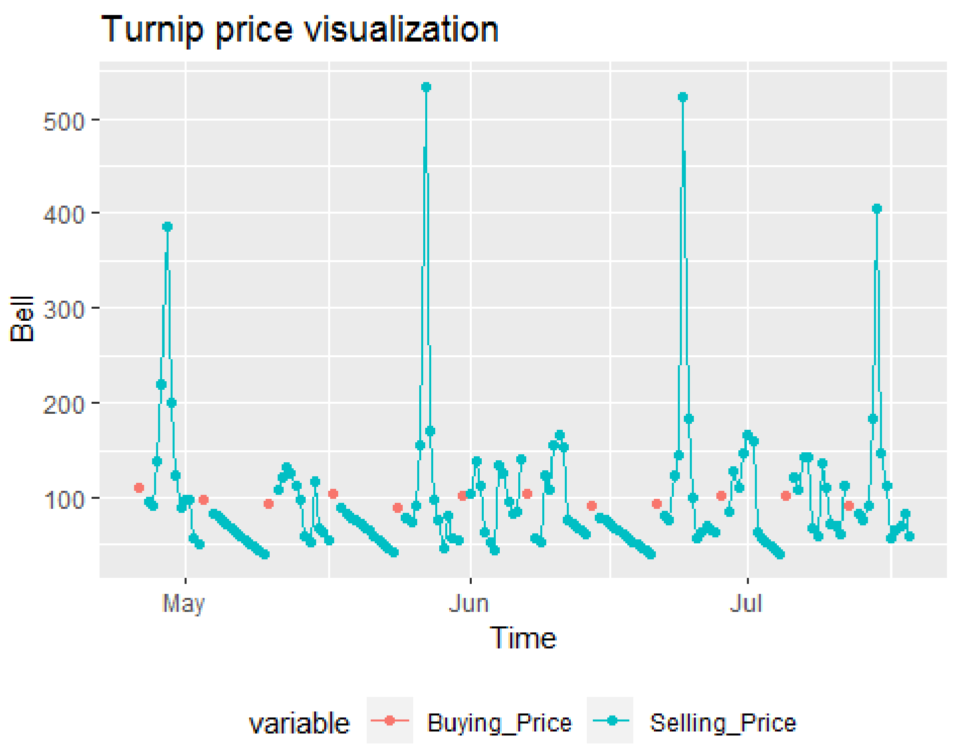
(a) Method: I choose k-nearest neighbor algorithm to impute the missing value of the selling prices on 5/1.

(b) Reason of using this method: If we try to impute the missing selling prices on 5/1 using the “hard” imputation with mean as follows

```
turnip_price_temp <- turnip_price
turnip_price_temp$Selling_Price <- impute(turnip_price$Selling_Price, mean)
turnip_price_imputed <- turnip_price
turnip_price_imputed[10:11, 3] <- turnip_price_temp[10:11, 3]
turnip_price_plot <- melt(turnip_price_imputed,  id.vars = 'Date_Time')
ggplot(turnip_price_plot, aes(x = Date_Time, y = value, colour = variable)) +
  geom_point(position = position_dodge(width = .3)) +
  geom_line(aes(colour = variable)) +
  labs(title = "Turnip price visualization", x = "Time", y = "Bell") +
  theme(legend.position="bottom")

## Warning: Removed 156 rows containing missing values (geom_point).

## Warning: Removed 13 row(s) containing missing values (geom_path).
```



then this will look unnatural to the rest of the data (see the above plot) because we have never seen anywhere that the selling price in the morning is equal to the selling price in the afternoon of the same day.

(c) R codes:

```
turnip_price_temp <- as.data.frame(turnip_price[, -2])
turnip_price_temp$Date_Time <-
as.integer(ifelse(hour(turnip_price_temp$Date_Time) == 0, 1, 2))
```

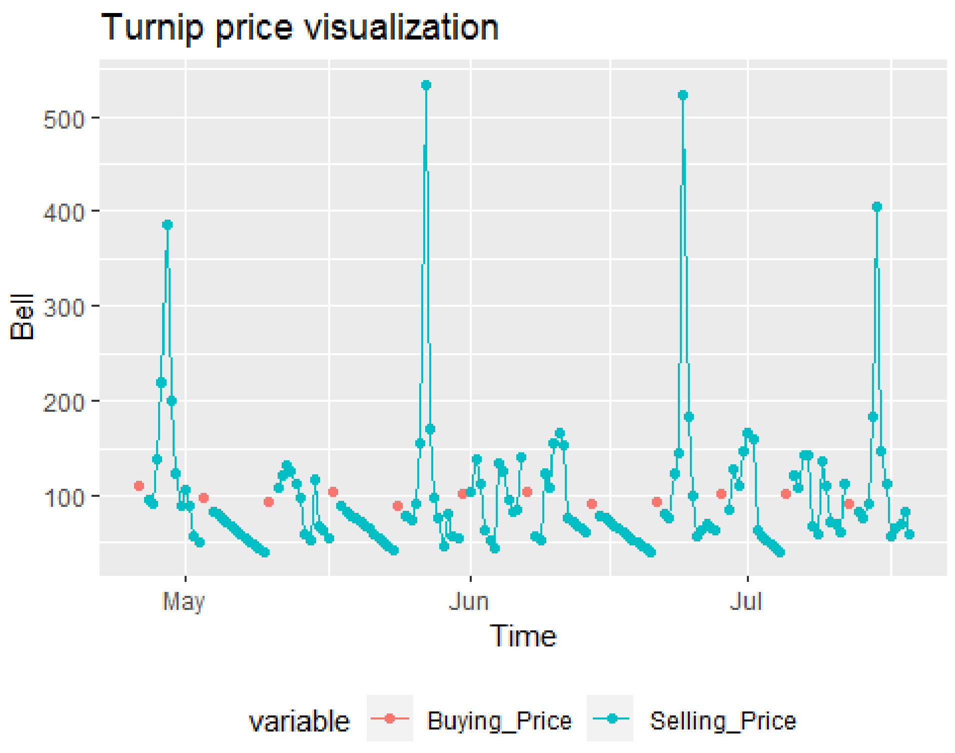
```r
turnip_price_temp <- complete(mice(turnip_price_temp, method =
"norm.predict", m = 1))
```

```
##
##  iter imp variable
##   1   1  Selling_Price
##   2   1  Selling_Price
##   3   1  Selling_Price
##   4   1  Selling_Price
##   5   1  Selling_Price
```

```r
turnip_price_imputed <- turnip_price
turnip_price_imputed[10:11, 3] <- turnip_price_temp[10:11, 2]
turnip_price_plot <- melt(turnip_price_imputed,  id.vars = 'Date_Time')
ggplot(turnip_price_plot, aes(x = Date_Time, y = value, colour = variable)) +
  geom_point(position = position_dodge(width = .3)) +
  geom_line(aes(colour = variable)) +
  labs(title = "Turnip price visualization", x = "Time", y = "Bell") +
  theme(legend.position="bottom")
```

```
## Warning: Removed 156 rows containing missing values (geom_point).
```

```
## Warning: Removed 13 row(s) containing missing values (geom_path).
```



(d) The results generated by the above R codes: in the report file.

(e) Summary of the results: In my result, the selling price in the morning (106.943662) of 5/1 is higher than the selling price in the afternoon of the same day (89.943662), which seems more natural than the imputation method using `mean` but it is still absurd to the whole original data. More precisely, in any week with highest peak like the week of 5/1, the selling price in the morning is always lower than the selling price in the afternoon (e.g., in the morning of 2020-05-29 the selling price was 47 which is lower than the selling price in the afternoon which was 80).

(f) Explanation of the results: This absurd behavior of my imputation is because I tried to impute the whole column of selling price including the Sundays where the players could not sell their white turnips. This means we have more predicted values than what we need, which probably caused the noise in predicting the selling prices on 5/1. Another reason could be because we use linear regression for method parameter of `mice` function. Besides, since it is impossible to have the actual selling prices on 5/1, I cannot see how good/bad `mice` function as well as imputation using `mean` worked in this case.

6. Probability of making profit:

(a) Method: I will compute every difference between the buying price on Sunday and each of the selling price in the morning and in the afternoon every weekday of the corresponding week. Then I calculate the probability of all possibilities where the computed difference is strictly positive, i.e. the probability of gaining profit from buying turnips.

(b) Reason of using this method: Because the player can only buy turnips on Sunday and have to sell them before the next Sunday, all the loss and profit must be calculated by using the buying price on Sunday and the selling prices on the weekdays of the same week as above. The rest of the method is just a usual way of calculating cumulative probability of a random variable achieving positive value.

(c) R codes:

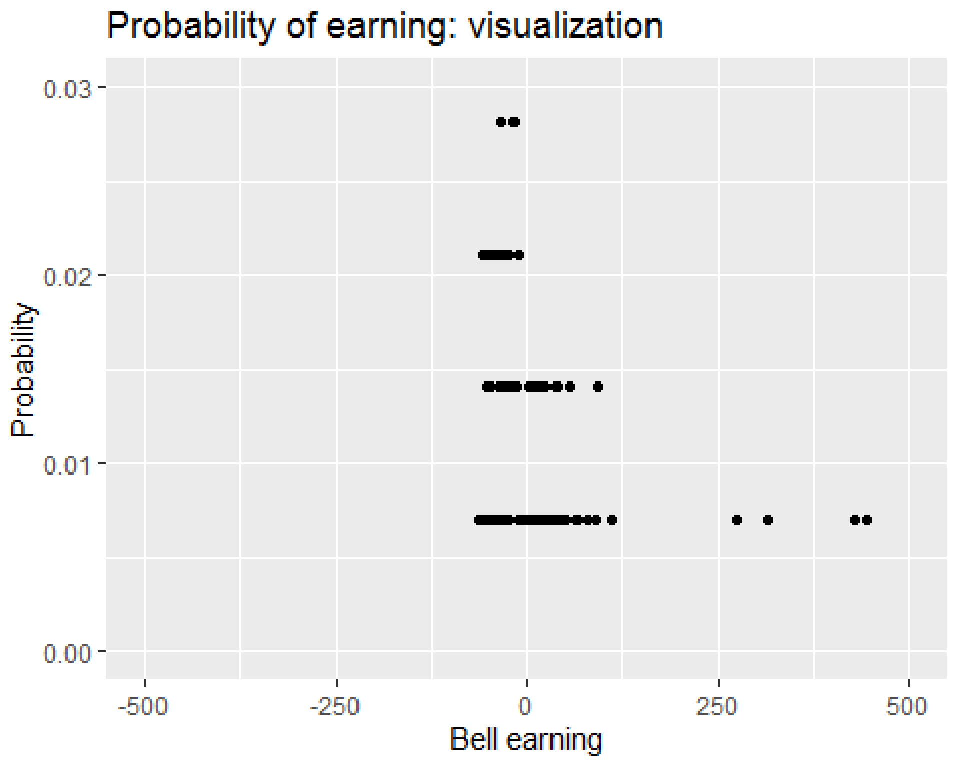
```r
earning <- 0
i <- 1
while (i <= nrow(turnip_price)) {
  if (wday(turnip_price[i, 1]) == 1) {
    for (j in c(1:12)) {
      if (is.na(turnip_price[i+j, 3]) == FALSE) {
        earning <- append(earning, turnip_price[i+j, 3] - turnip_price[i, 2])
      }
    }
    i <- i + 13
  } else {
    i <- i + 1
  }
}
earning <- earning[-1]
p_pos_earning <- round(length(which(earning>0))/length(earning)*100, digits =
2)
print(p_pos_earning)
```

```
## [1] 34.51
```

Here is the graph visualizing the probability of gaining or losing bells based on the given data.

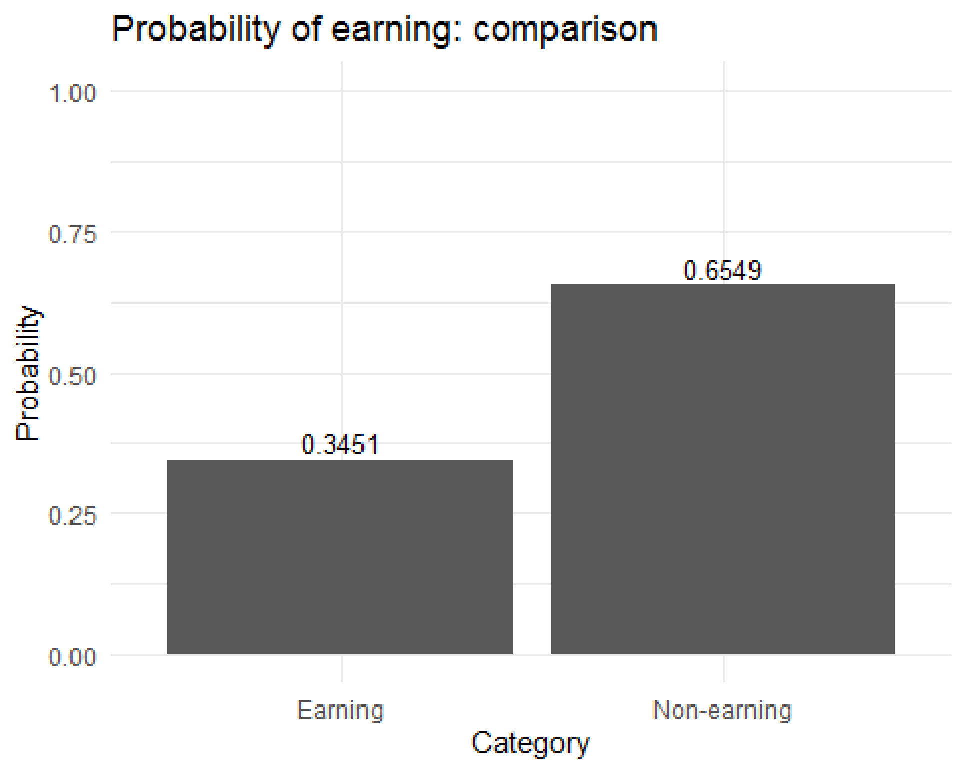
```
# earning_df <- c(min(earning, na.rm = TRUE):max(earning, na.rm = TRUE))
earning_df <- sort(earning, decreasing = FALSE)
earning_df <- as.data.frame(earning_df)
colnames(earning_df) <- c("Earning")

earning_df <- earning_df %>%
  distinct() %>%
  mutate(Prob = 0*Earning)
for (i in c(1:nrow(earning_df))) {
  earning_df$Prob[i] <- length(which(earning ==
earning_df$Earning[i]))/length(earning)
}
ggplot(data = earning_df, aes(x = Earning, y = Prob)) +
  xlim(-500, 500) +
  ylim(0, 0.03) +
  geom_point() +
  labs(title = "Probability of earning: visualization", x = "Bell earning", y
= "Probability") +
  theme(legend.position="bottom")
```



Also, this is a comparison between the cumulative probability of earning profit and that of non-earning profit.

```r
earning_df <- c("Non-earning", "Earning")
earning_df <- as.data.frame(earning_df)
colnames(earning_df) <- c("Category")
earning_df <- earning_df %>%
  mutate(Prob = c(1-p_pos_earning/100, p_pos_earning/100))
ggplot(data = earning_df, aes(x = Category, y = Prob)) +
  ylim(0, 1) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = Prob), vjust = -0.3, size = 3.5) +
  theme_minimal() +
  labs(title = "Probability of earning: comparison", x = "Category", y =
"Probability") +
  theme(legend.position="bottom")
```



(d) The result generated by the above R code: The probability of earning money (bell) from buying white turnips in this game is approximately 34.51%.

(e) Summary of the result: The chance of gaining profit from buying turnips and selling them within one week is only higher one third a little bit which is not a high probability. Hence, as a consequence, the players will be more likely to loose their money (bells) according to the given data.

(f) Explanation of the result: Just by looking at the graph in the first question, we can easily see that though sometimes the selling price grew steeply, the number of times when the selling price dropped down to a place lower than or equal to the buying price on Sunday of that week is much higher than the number of times when the opposite event happened.Therefore, it is natural for us to have the result obtained earlier. Besides, the comparison plot in part (c) helps us see more clearly what I have just stated.

7. Recognizing the patterns of the given data:

(a) Method: I will create a dataframe containing the fluctuations of selling prices and name it `fluctuation_df`. After that, k-means clustering algorithm will be applied to the `fluctuation_df` dataframe below to classify all its observations to their appropriate groups.

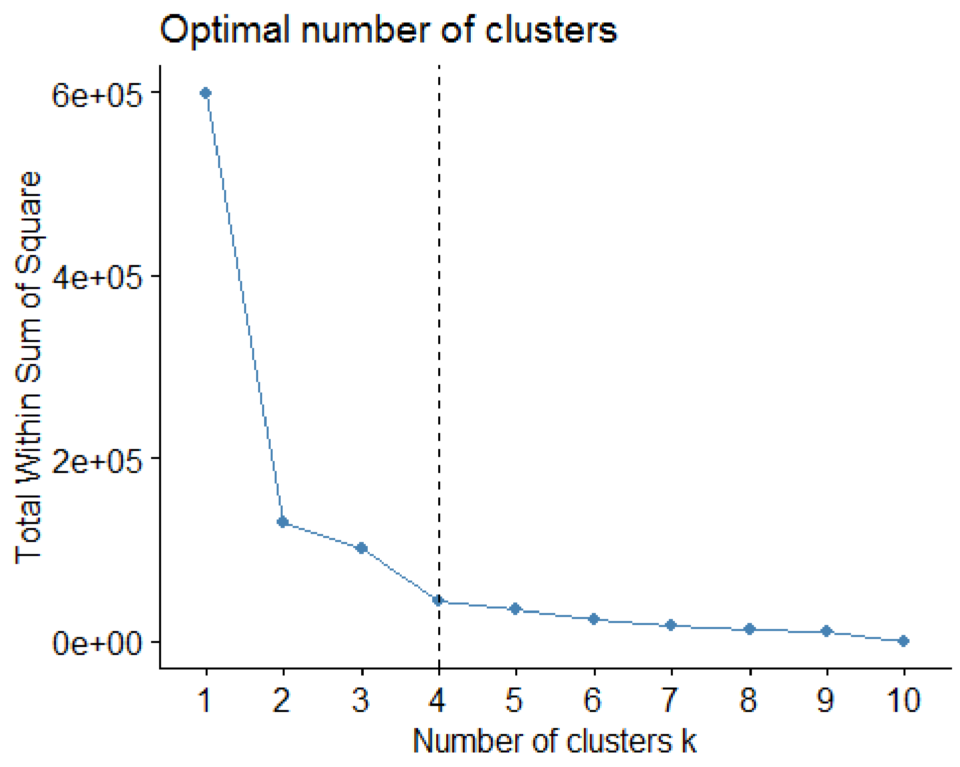
(b) Reason of using this method: The problem of finding possible patterns shown up in a dataframe is equivalent to clustering similar observations into the same groups and I am more familiar with k-means clustering algorithm than other algorithms. These are why I chose this method.

(c) R codes: Since there are 2 missing selling prices on 5/1, I will use the imputed data from question 5 for the sake of computation to create the following data. Notice that we are working on the selling price of turnip which means all columns have the same unit, so there is no need for scaling this dataframe.

```r
# create fluctuation_df
nSun <- length(which(wday(turnip_price_imputed$Date_Time) == 1))
fluctuation_df <- numeric(nSun)
fluctuation_df <- as.data.frame(fluctuation_df)
colnames(fluctuation_df) <- c("MM")
fluctuation_df <- fluctuation_df %>%
  mutate(MT = MM) %>%
  mutate(TT = MM) %>%
  mutate(TW = MM) %>%
  mutate(WW = MM) %>%
  mutate(WR = MM) %>%
  mutate(RR = MM) %>%
  mutate(RF = MM) %>%
  mutate(FF = MM) %>%
  mutate(FS = MM) %>%
  mutate(SS = MM)
i <- 0
while (i < nSun) {
  i <- i + 1
  j <- i + (i-1)*12
  for (k in c(1:11)) {
    fluctuation_df[i, k] <- turnip_price_imputed$Selling_Price[j+k+1] -
turnip_price_imputed$Selling_Price[j+k]
  }
}

# compute the optimal number of clusters and apply k-means algorithm to
fluctuation_df
```

```r
fviz_nbclust(fluctuation_df, kmeans, method = "wss") +
  geom_vline(xintercept = 4, linetype = 2)
```



```r
set.seed(123)
fluctuation_clustered <- kmeans(fluctuation_df, 4, nstart = 25)
fluctuation_df <- cbind(fluctuation_df, cluster =
fluctuation_clustered$cluster)
print(fluctuation_df)
```

```
##      MM  MT  TT  TW   WW  WR  RR        RF  FF        FS  SS cluster
## 1    -4  46  82 165 -184 -78 -33  16.94366 -17 -31.94366  -6       2
## 2    -4  -4  -4  -4   -5  -3  -5  -4.00000  -3  -4.00000  -4       3
## 3    13  10  -6 -13  -14 -39  -6  64.00000 -48  -6.00000  -7       3
## 4    -5  -5  -3  -4   -4  -3  -5  -4.00000  -5  -4.00000  -5       3
## 5    -4  17  64 378 -364 -73 -20 -30.00000  33 -23.00000  -2       4
## 6    34 -26 -50 -10   -9  91  -9 -31.00000 -13   3.00000  55       1
## 7    -4  71 -15  47   11 -13 -77  -4.00000  -4  -4.00000  -4       3
## 8    -3  -3  -4  -3   -4  -4  -4  -4.00000  -3  -3.00000  -4       3
## 9    -4  46  21 379 -340 -83 -43   7.00000   7  -6.00000  -1       4
## 10   42 -17  36  20   -8 -96  -5  -5.00000  -4  -4.00000  -5       3
## 11  -12  34   0 -74   -9  76 -25 -39.00000  -1  -9.00000  51       1
## 12   -5  14  92 222 -257 -36 -55   8.00000   6  11.00000 -23       2
```

(d) The result generated by the above R code: There is clearly an “elbow” at `k = 4`, which indicates any cluster beyond the fourth has little value. Therefore, the optimal number of clusters must be 4. Also, I have also classified all 12 observations in `fluctuation_df` into 4 clusters as shown in the above R codes.

(e) Summary of the result: Again, we have 4 clusters. In pattern 1, we have the 6th and 11th weeks. In pattern 2, the first and the last weeks have smaller peaks than the 5th and 9th weeks which belong to pattern 4. In the last pattern - pattern 3, the 2nd, the 4th and the 8th weeks clearly have the same decreasing patterns. Meanwhile, the 3rd, the 7th, and the 10th weeks do not seem to be in the same group of the 2nd, the 4th, and the 8th weeks, but they actually do.

(f) Explanation of the result: It is pretty clear from the plot why the first and the last weeks belong to pattern 2 and the 5th and 9th weeks belong to pattern 4. The reason for why the 3rd, the 7th, and the 10th weeks belong to the same group with the 2nd, the 4th, and the 8th weeks is that from Friday's afternoon the selling prices are lower than the buying price of Sunday in the same week and hence the players will lose their money (bells) if they choose to sell their turnips. In pattern number 1, the common point of the 6th and 11th weeks is that there are 3 increases in selling price: the first one happens at the beginning of the week, the second one happens in the middle of the week, and the last one happens in Saturday's afternoon.

8. Conditional probability of a pattern occurring given that another pattern has occurred:

(a) Method: Assume we want to find the likelihood of pattern X this week given the occurrence of pattern Y last week. Based on the current occurrences of patterns in the result of the previous question, the conditional probability is equal to the frequency of pattern X occurring after pattern Y divided by the frequency of an arbitrary pattern in the above 4 patterns occurring after pattern Y.

(b) Reason of using this method: In my understanding, question 8 is equivalent to the problem of finding the conditional probability of a pattern occurring given that another pattern has occurred, so it is natural to use the above method.

(c) R codes: Assume after the 12th observation in `fluctuation_df`, the cluster of the 13th observation is not 2.

```r
prob_tbl <- matrix(numeric(16), ncol = 4)
colnames(prob_tbl) <- c('1', '2', '3', '4')
rownames(prob_tbl) <- c('1', '2', '3', '4')
prob_tbl <- as.table(prob_tbl)

for (i in c(1:4)) {
  pSample <- which(fluctuation_df$cluster == i)
  nSample <- length(pSample)
  pSample <- pSample + 1
  for (j in c(1:4)) {
    nFreq <- 0
    for (k in c(1:nSample)) {
      if (pSample[k] <= 12 && fluctuation_df$cluster[pSample[k]] == j) {
        nFreq <- nFreq + 1
      }
    }
    prob_tbl[as.character(j), as.character(i)] <- nFreq/nSample
  }
}

print(prob_tbl)
```

```
##           1         2         3         4
## 1 0.0000000 0.0000000 0.1666667 0.5000000
## 2 0.5000000 0.0000000 0.0000000 0.0000000
## 3 0.5000000 0.5000000 0.5000000 0.5000000
## 4 0.0000000 0.0000000 0.3333333 0.0000000
```

(d) The result generated by the above R code: Based on the `fluctuation_df` dataframe, we have the above table containing all the probability of the pattern labeled by the table's row name occurring given the pattern labeled by the table's column name has occurred.

(e) Summary of the result: If the pattern of this week is 1, there is a 50-50 chance that in the next week pattern 2 or 3 will show up. Under our assumption, there is 50% pattern 3 will follow pattern 2. Next, following pattern 3 will be either pattern 1 (16.67%), another pattern 3 (50%), or pattern 4 (33.33%). Now if we look at the table another way, there is 50% pattern 3 will occur after any pattern shows up right before it. Besides, pattern 2 and pattern 4 will only occur after pattern 1 and pattern 3, respectively. Overall, the probabilities seem low (16.67% or 33.33%) or undecidable (50%).

(f) Explanation of the result: The reason why the probability table above seems lack of information (many zero probabilities and most of nonzero probabilities is 50%) is because the data of patterns is not large enough to give us a more precise and decidable prediction of all conditional probabilities we want to see.

9. Recognizing the patterns of the given data:

(a) Method: I will calculate the probability to be an outlier of everyday from Monday to Saturday. The algorithm used to test whether a low or a high value is an outlier in this section will be Dixon's test.

(b) Reason of using this method: The reason of using Dixon's test here is because of the small size of our samples (data of selling prices in 6 days) and Dixon's test is guaranteed to be most useful in this case.

(c) R codes:

```
highest <- numeric(7)
lowest <- numeric(7)
i <- 0
while (i < nSun) {
  i <- i + 1
  j <- i + (i-1)*12
  subturnip_price <- turnip_price_imputed[(j+1):(j+12), ]
  test <- dixon.test(subturnip_price$Selling_Price)
```

```r
  if (test$p.value < 0.05) {
    print(i)
    k <- wday(subturnip_price$Date_Time[which(subturnip_price$Selling_Price
== max(subturnip_price$Selling_Price))])
    highest[k] <- highest[k] + 1
  }
  test <- dixon.test(subturnip_price$Selling_Price, opposite = TRUE)
  if (test$p.value < 0.05) {
    print(i)
    k <- wday(subturnip_price$Date_Time[which(subturnip_price$Selling_Price
== min(subturnip_price$Selling_Price))])
    lowest[k] <- lowest[k] + 1
  }
}
```

```
## [1] 5
## [1] 9
## [1] 9
## [1] 12
```

```r
highest <- highest/12
lowest <- lowest/12
print(highest)
```

```
## [1] 0.00 0.00 0.00 0.25 0.00 0.00 0.00
```

```r
print(lowest)
```

```
## [1] 0.00000000 0.00000000 0.00000000 0.00000000 0.08333333 0.00000000
0.00000000
```

(d) The result generated by the above R code: the probability of some certain days on which the selling price tends to be higher or lower than on other days.

(e) Summary of the results: There is a 25% of the selling prices on Wednesdays are significantly higher than the selling prices on other days. Similarly, there is an 8.33% of the selling prices on Thursdays are significantly lower than the selling prices on other days.

(f) Explanation of the results: It is not difficult to see why a quarter of selling prices on Wednesdays are significantly higher the selling prices on other days. 4 out of 12 Wednesdays already contain all 4 peak selling prices. The reason for why there is about 8% of selling prices on Thursdays are significantly lower than the selling prices on other days is quite vague. It is probably because the peak selling price of 9th week is too high, which made the mean higher and left the minimum selling price on Thursday as an outlier.

10. Prediction of the selling prices in the later half of a week in 2 scenarios:

(a) Method:

(b) Reason of using this method:

(c) R codes:

(d) The result generated by the above R code:

(e) Summary of the result:

(f) Explanation of the result: