Intro to Data Science Final Project

Christopher Rutherford

Fall 2020

Overview

In this project, we will analyze the Animal Crossing Stalk Market prices from April 26, 2020 through July 18, 2020. The Stalk market provides selling prices for turnip prices every week. Turnips can be bought every Sunday before noon. Starting Monday morning, turnip prices are appraised twice a day throughout the week, once in the morning (before 12pm) and again in the afternoon. The goal is to maximize profits or, at the very least, minimize losses.

To make our work more streamlined with `ggplot2` and `dplyr`, we will convert our data frame to a tibble. Additionally, we'll add a column to indicate the day of week, which will help us with our calculations and visualizations throughout this analysis.

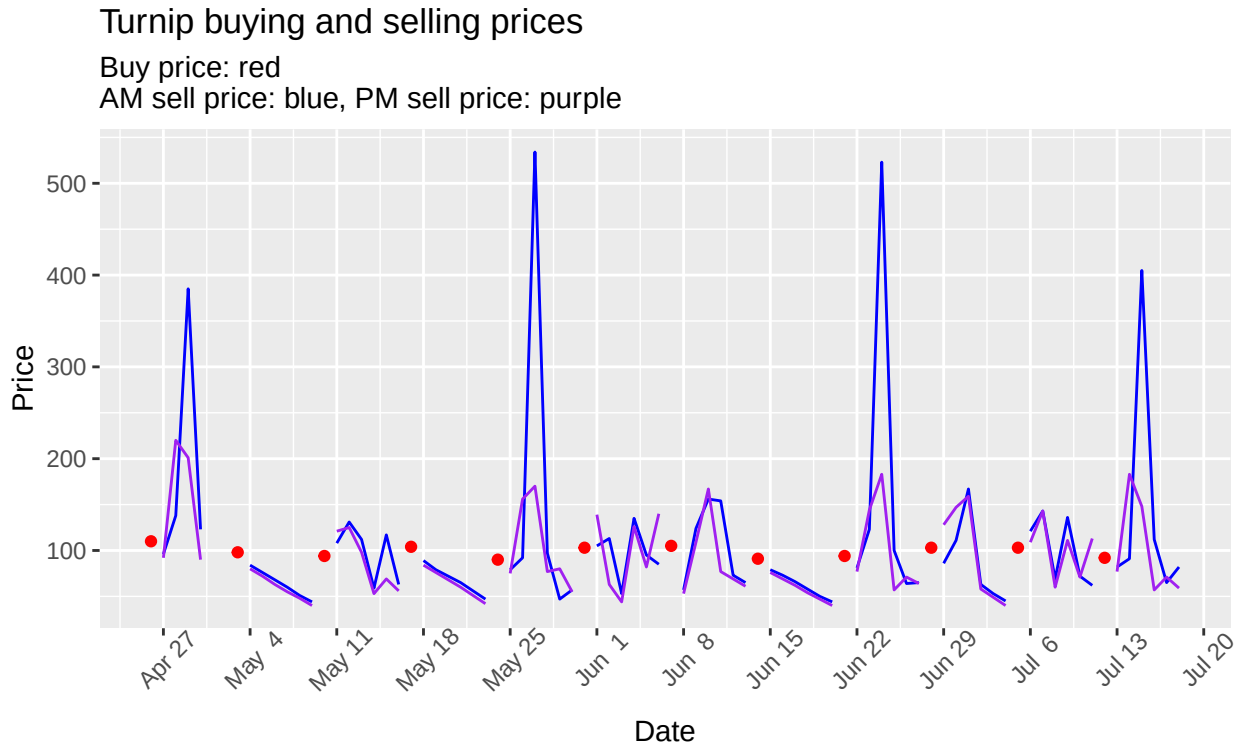
```
turnips = as_tibble(turnips)
turnips = turnips %>%
  mutate(weekday = factor(x = weekdays(date),
                          levels = c("Sunday", "Monday", "Tuesday", "Wednesday",
                                     "Thursday", "Friday", "Saturday"))) %>%
  mutate(date = as.Date(date))
turnips
```

```
## # A tibble: 84 x 5
##    date       buy_price sell_price_am sell_price_pm weekday
##    <date>         <dbl>         <dbl>         <dbl> <fct>
##  1 2020-04-26       110            NA            NA Sunday
##  2 2020-04-27        NA            96            92 Monday
##  3 2020-04-28        NA           138           220 Tuesday
##  4 2020-04-29        NA           385           201 Wednesday
##  5 2020-04-30        NA           123            90 Thursday
##  6 2020-05-01        NA            NA            NA Friday
##  7 2020-05-02        NA            58            52 Saturday
##  8 2020-05-03        98            NA            NA Sunday
##  9 2020-05-04        NA            84            80 Monday
## 10 2020-05-05        NA            76            72 Tuesday
## # ... with 74 more rows
```

1. Visualizing the data set

Since this data is set up as a time series, it is best to visualize this with a scatter/line plot. In particular, we'll plot the date along the x-axis and turnip prices along the y-axis. We will use dots for the buying price since they only appear once a week. Including the buying price will give us a better idea of how much turnip prices change after buying them.

```
ggplot(turnips, aes(x = date))+
  geom_point(aes(y =  buy_price), color = 'red', na.rm = T)+
  geom_line(aes(y = sell_price_am), color = 'blue', na.rm = T)+
  geom_line(aes(y = sell_price_pm), color = 'purple', na.rm = T)+
  labs(x = "Date", y = "Price",
       title = "Turnip buying and selling prices",
       subtitle = "Buy price: red\nAM sell price: blue, PM sell price: purple")+
  scale_x_date(breaks = "1 week", date_labels = "%b %e")+
  theme(axis.text.x = element_text(angle = 45, vjust = 0.8))
```



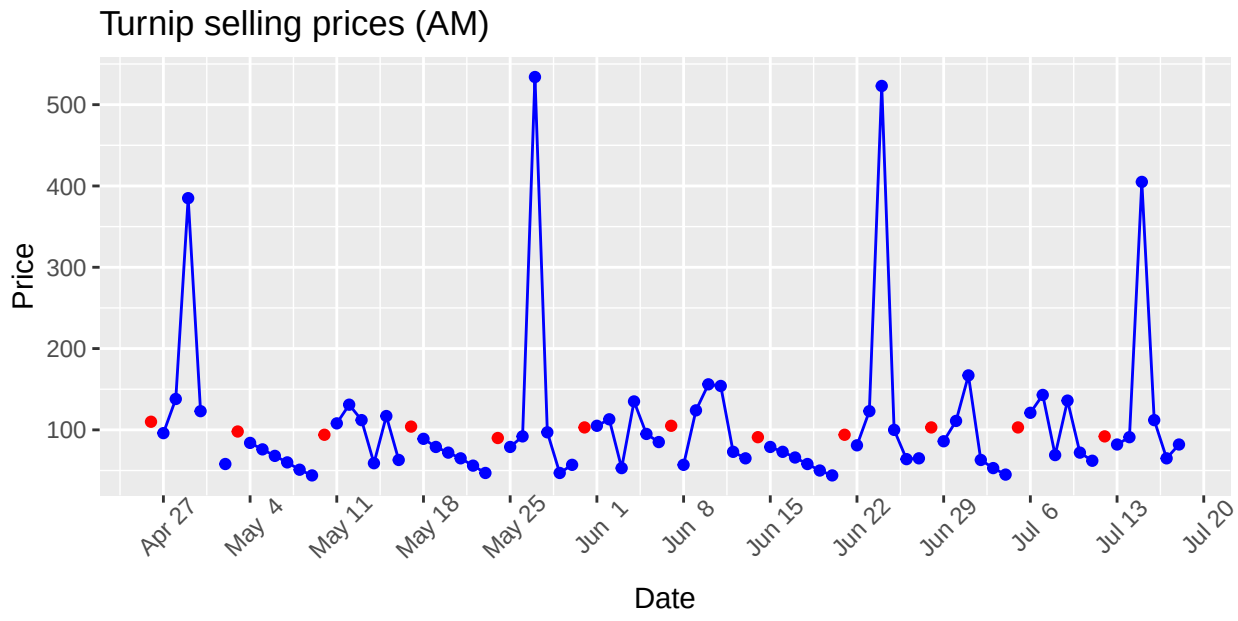
While this is certainly not the cleanest plot to look at, there are a few observations and other things to take note of:

- Turnips are sold on Sundays and prices are not appraised on Sundays, causing the repetitive breaks in the blue and purple lines
- May 1 has no selling price due to the Stalk market being unavailable
- Morning selling prices reach much higher peaks than afternoon selling prices
- Afternoon selling prices are rarely higher than that same day's morning selling price

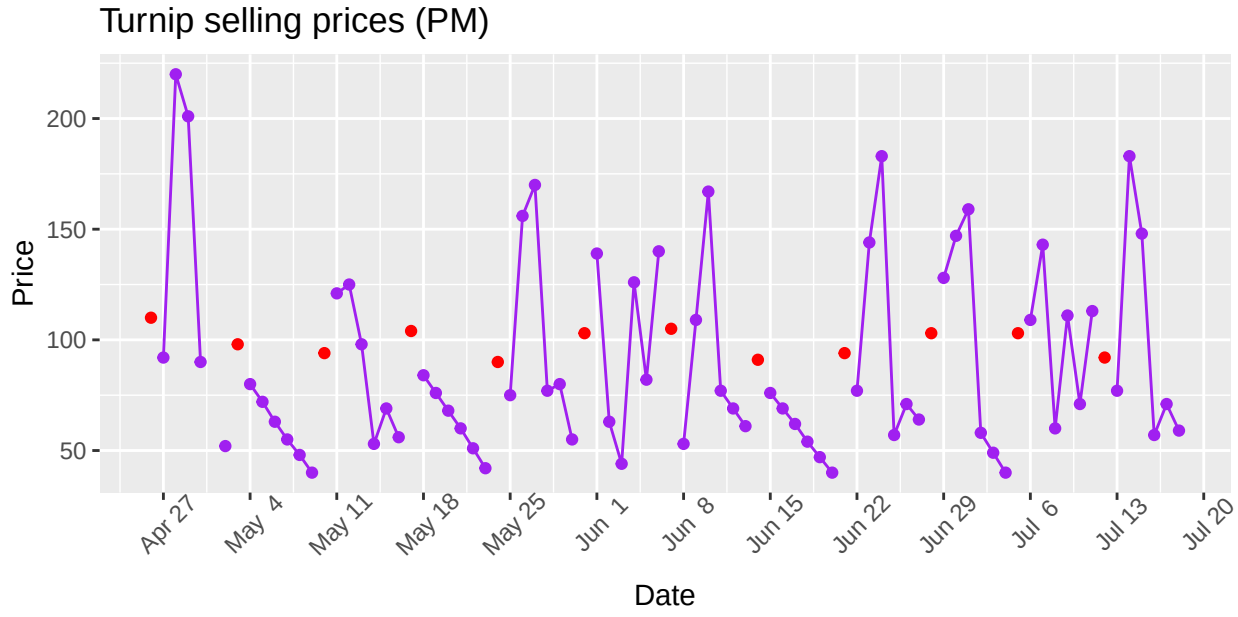
We'll look at each set (AM/PM) of selling prices in separate graphs. Like the previous plot, points for the the buying prices will be included.

```
ggplot(turnips, aes(date, sell_price_am))+
  geom_line(color = 'blue', na.rm = T)+
  geom_point(color = 'blue', na.rm = T)+
  geom_point(aes(y = buy_price), color = 'red', na.rm = T)+
  labs(x = "Date", y = "Price", title = "Turnip selling prices (AM)")+
```

```
  scale_x_date(date_breaks = "1 week", date_labels = "%b %e")+
  theme(axis.text.x = element_text(angle = 45, vjust = 0.8))
```




```
ggplot(turnips, aes(date, sell_price_pm))+
  geom_line(color = 'purple', na.rm = T)+
  geom_point(color = 'purple', na.rm = T)+
  geom_point(aes(y = buy_price), color = 'red', na.rm = T)+
  labs(x = "Date", y = "Price", title = "Turnip selling prices (PM)")+
  scale_x_date(date_breaks = "1 week", date_labels = "%b %e")+
  theme(axis.text.x = element_text(angle = 45, vjust = 0.8))
```



We can see from these plots that AM selling prices can peak at over 400 and even 500 bells, while PM selling prices can peak around 150-200 bells. For the most part, the prices seem to peak on Tuesdays and Wednesdays. However, in certain instances, selling prices remain lower than the buying price for the entire week.

2. Mean and range of the buying price

Since turnips can only be bought on Sundays, we can filter out the other six days and manually calculate the mean and range of the buying price.

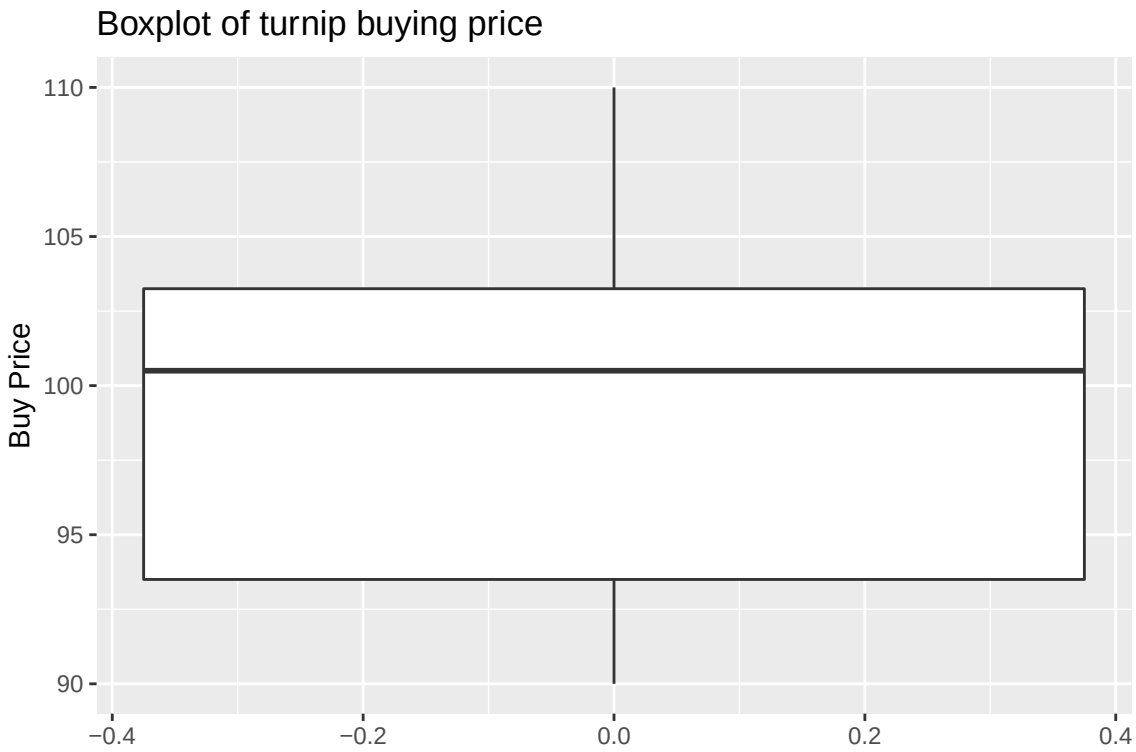
```
turnips %>%
  filter(weekday == "Sunday") %>%
  select(buy_price, weekday) %>%
  summarise(avg_buy_price = mean(buy_price),
            range_buy_price = range(buy_price))
```

```
## # A tibble: 2 x 2
##   avg_buy_price range_buy_price
##           <dbl>           <dbl>
## 1          98.9              90
## 2          98.9             110
```

So, on average, turnips can be bought for about 99 bells each Sunday, with the price ranging from 90 bells to 110 bells. Since the average is a little closer to 90, we may expect bells to have higher prices less often.

Additionally, we can get a better idea of the distribution of buying prices with a boxplot:

```
turnips %>%
  filter(weekday == "Sunday") %>%
  ggplot(aes(y = buy_price))+
  geom_boxplot()+
  labs(y = "Buy Price", title = "Boxplot of turnip buying price")
```



As we can see from the boxplot, buying prices are somewhat left skewed - the median is 100.5, while the mean is 98.917.

3. Mean and range of the selling price

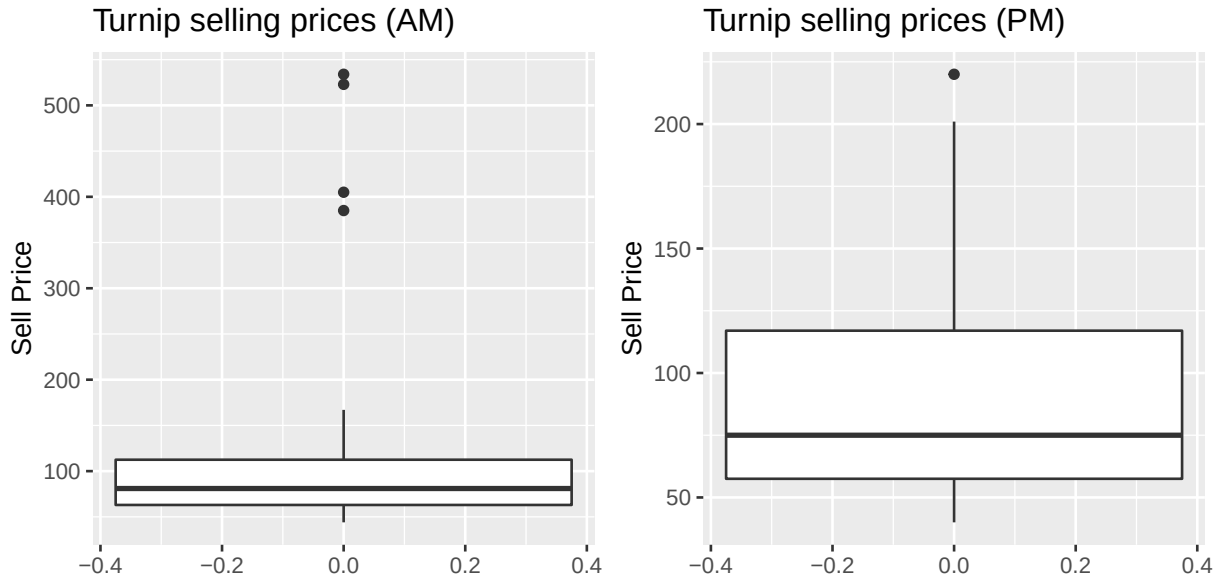
Like the buying prices, we'll calculate the mean and range manually and visualize the distributions of the selling prices. Visualizing the distribution can help identify possible outliers and explain the differences in the means and ranges.

```
turnips %>%
  filter(weekday != "Sunday") %>%
  select(sell_price_am, sell_price_pm, weekday) %>%
  summarise(sell_price_am_avg = mean(sell_price_am, na.rm = T),
            sell_price_pm_avg = mean(sell_price_pm, na.rm = T),
            sell_price_am_range = range(sell_price_am, na.rm = T),
            sell_price_pm_range = range(sell_price_pm, na.rm = T))
```

```
## # A tibble: 2 x 4
##   sell_price_am_avg sell_price_pm_avg sell_price_am_range sell_price_pm_range
##               <dbl>             <dbl>               <dbl>               <dbl>
## 1              107.              89.9                  44                  40
## 2              107.              89.9                 534                 220
```

We can see that morning selling prices are, on average, higher than afternoon selling prices. Additionally, the range of morning selling prices is much larger, with AM selling prices ranging from 44 bells to 534 bells, a range of 490, and PM selling prices ranging from 40 to 220, a range of only 180. We will create boxplots for these prices as well.

```
ggplot(turnips)+
  geom_boxplot(aes(y = sell_price_am), na.rm = T)+
  labs(y = "Sell Price", title = "Turnip selling prices (AM)")+
ggplot(turnips)+
  geom_boxplot(aes(y = sell_price_pm), na.rm = T)+
  labs(y = "Sell Price", title = "Turnip selling prices (PM)")
```



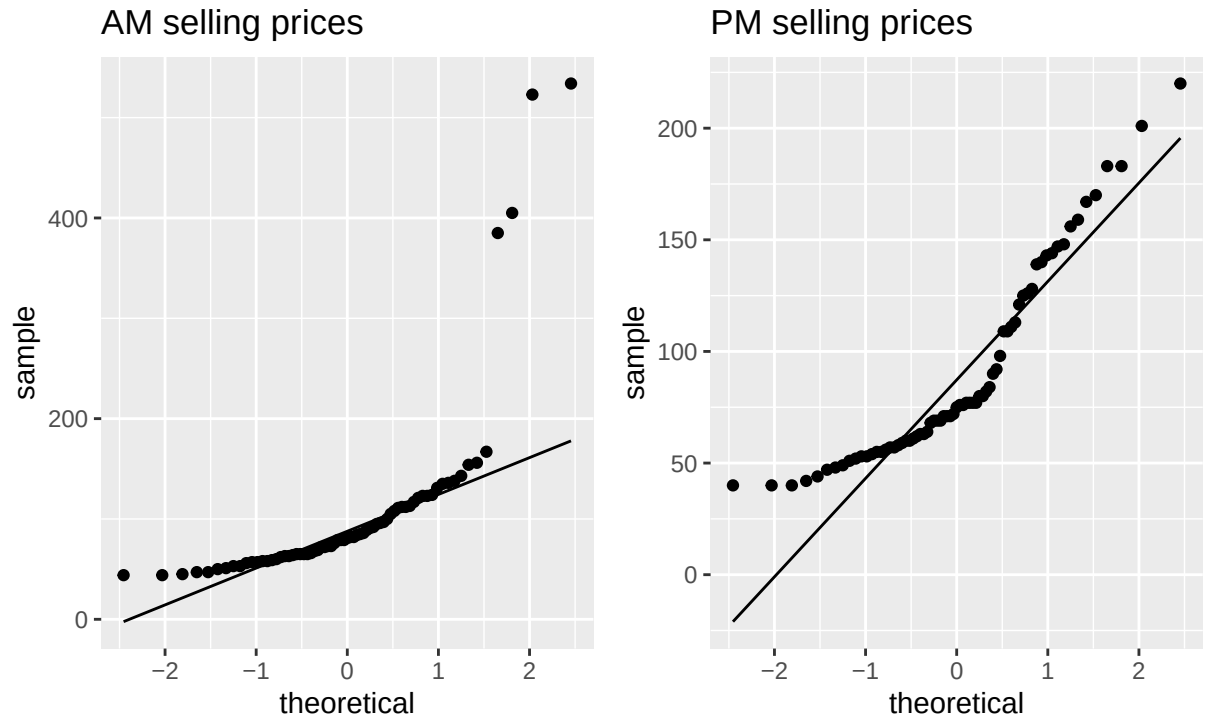
We can see that both sets of selling prices have a median price of about 75 bells. However, the AM selling price has several more outliers that are much larger than the PM selling prices, giving it a much larger skew than the PM selling price.

4. Morning vs. afternoon turnip selling prices

While answering the first three questions, we discovered that morning selling prices are typically higher than afternoon selling prices. But just how significant is this difference? How consistent is this difference? We have a few ways to approach this question, particularly through statistical tests and to compare the averages for each set of selling prices.

We can investigate the normality of the selling prices to determine which test may be more appropriate for this data. Performing a t-test on data that exhibits possible non-normality can give us misleading results [1].

```
ggplot(turnips, aes(sample = sell_price_am))+
  stat_qq(na.rm = T)+
  stat_qq_line(na.rm = T)+
  labs(title = "AM selling prices")+
ggplot(turnips, aes(sample = sell_price_pm))+
  stat_qq(na.rm = T)+
  stat_qq_line(na.rm = T)+
  labs(title = "PM selling prices")
```



The large outliers in the AM selling prices are very apparent on the Q-Q plot. The PM selling price outliers are not nearly as extreme by comparison, but the lack of a linear trend is quite apparent. The Mann-Whitney test may be more appropriate for this data than a t-test, as it does not require data to follow a normal distribution and is more robust.

```
wilcox.test(x = turnips$sell_price_am,
            y = turnips$sell_price_pm, conf.int = T)
```

```
##
```

```
##  Wilcoxon rank sum test with continuity correction
##
## data:  turnips$sell_price_am and turnips$sell_price_pm
## W = 2726.5, p-value = 0.4017
## alternative hypothesis: true location shift is not equal to 0
## 95 percent confidence interval:
##  -6.000041 14.999963
## sample estimates:
## difference in location
##               4.000016
```

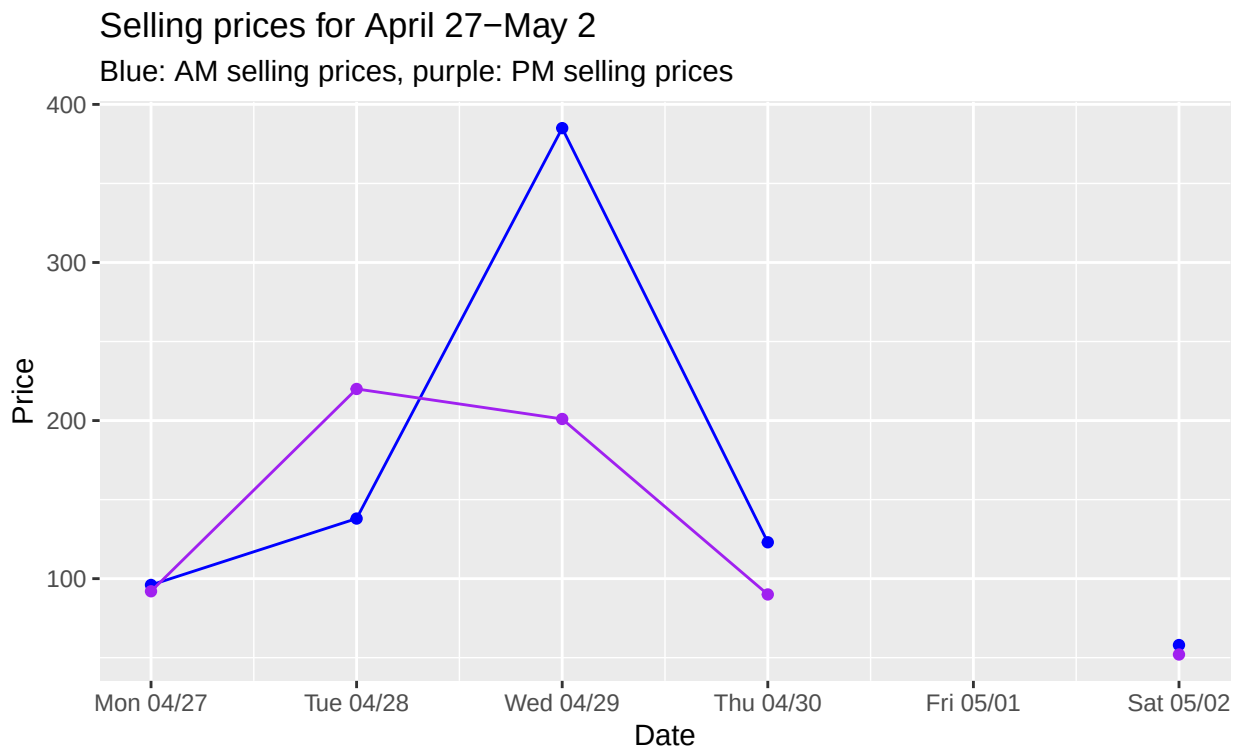
With the data we have, we cannot conclusively claim that AM selling prices are statistically significantly higher than the PM selling prices (p-value ≈ 0.4).

5. Dealing with the missing prices

Because the Stalk market was closed on May 1st, there is no selling price data provided. However, we'd like to replace these missing values with reasonable and probable selling prices for this day.

We can begin by plotting the prices for the week with missing selling prices and analyzing the pattern to compare it to other weeks.

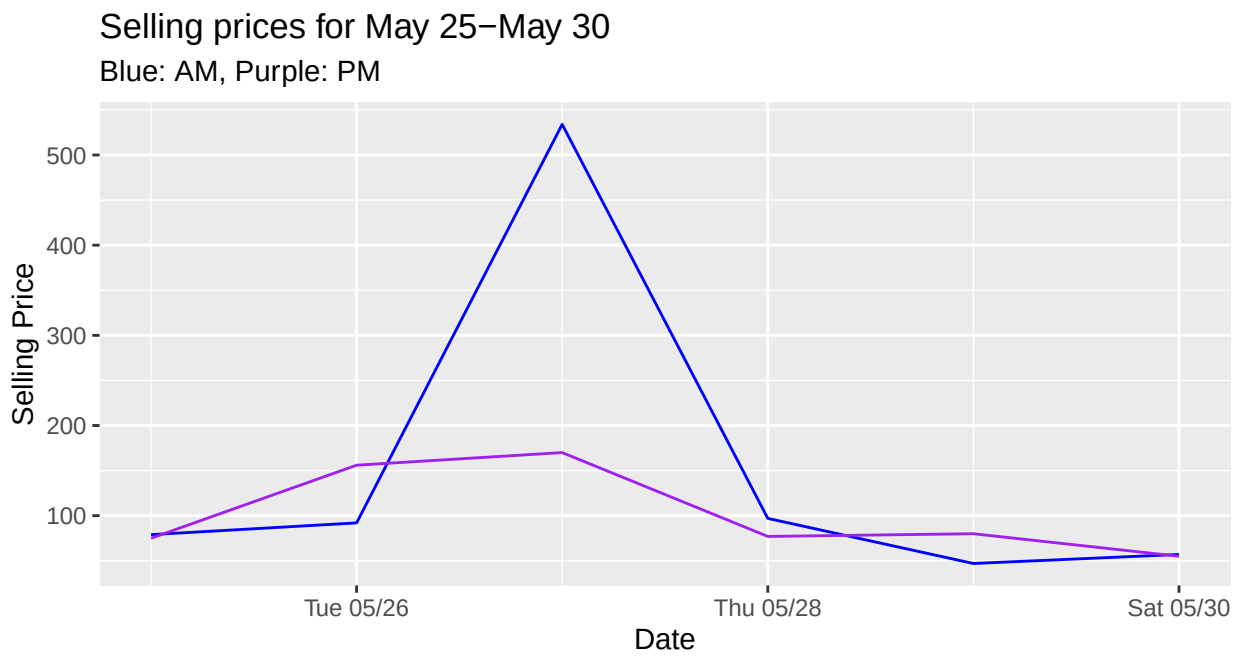
```
turnips %>%
  filter(date <= "2020-05-02" & date > "2020-04-26") %>%
  ggplot(aes(x = date))+
  geom_line(aes(y = sell_price_am), na.rm = T, color = 'blue')+
  geom_point(aes(y = sell_price_am), na.rm = T, color = 'blue')+
  geom_line(aes(y = sell_price_pm), na.rm = T, color = 'purple')+
  geom_point(aes(y = sell_price_pm), na.rm = T, color = 'purple')+
  scale_x_date(date_labels = "%a %m/%d", date_breaks = "day")+
  labs(x = "Date", y = "Price", title = "Selling prices for April 27-May 2",
       subtitle = "Blue: AM selling prices, purple: PM selling prices")
```



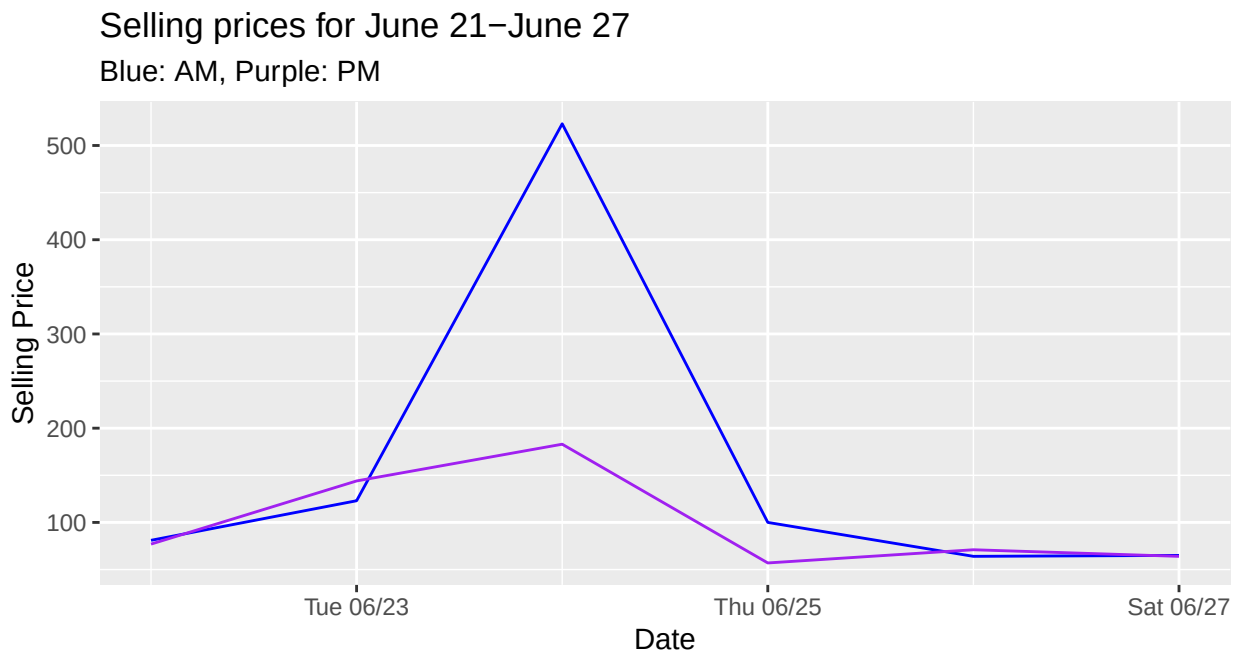
Based on this plot, we can see that selling prices for this week have a large spike on Wednesday, a pattern that occurs three other times in this data set. We can look at the prices for those weeks to determine what Friday's selling prices usually are. In particular, we want the prices for the weeks starting 5/24, 6/21, and 7/12. We will plot the selling prices each of these weeks:

```
turnips %>%
  filter(date > "2020-05-24" & date <= "2020-05-30") %>%
  ggplot(aes(x = date))+
  geom_line(aes(y = sell_price_am), na.rm = T, color = 'blue')+
  geom_line(aes(y = sell_price_pm), na.rm = T, color = 'purple')+
  scale_x_date(date_labels = "%a %m/%d")+
```

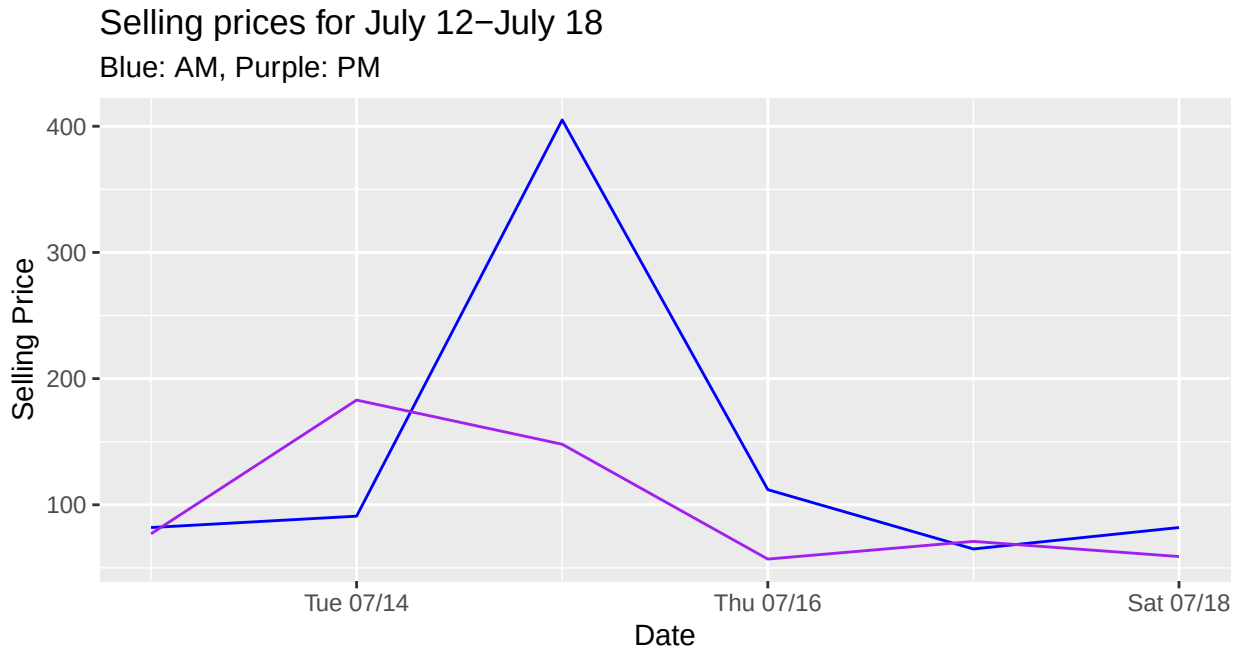
```
  labs(title = "Selling prices for May 25-May 30",
       subtitle = "Blue: AM, Purple: PM",
       x = "Date", y = "Selling Price")
```




```
turnips %>%
  filter(date > "2020-06-21" & date <= "2020-06-27") %>%
  ggplot(aes(x = date))+
  geom_line(aes(y = sell_price_am), na.rm = T, color = 'blue')+
  geom_line(aes(y = sell_price_pm), na.rm = T, color = 'purple')+
  scale_x_date(date_labels = "%a %m/%d")+
  labs(title = "Selling prices for June 21-June 27",
       subtitle = "Blue: AM, Purple: PM",
       x = "Date", y = "Selling Price")
```




```
turnips %>%
  filter(date > "2020-07-12" & date <= "2020-07-18") %>%
  ggplot(aes(x = date))+
  geom_line(aes(y = sell_price_am), na.rm = T, color = 'blue')+
  geom_line(aes(y = sell_price_pm), na.rm = T, color = 'purple')+
  scale_x_date(date_labels = "%a %m/%d")+
  labs(title = "Selling prices for July 12-July 18",
       subtitle = "Blue: AM, Purple: PM",
       x = "Date", y = "Selling Price")
```



Based on these plots, Friday's AM selling prices are usually lower than Thursday and Saturday's AM selling

prices, but Friday's PM selling prices are usually higher than Thursday and Saturday's PM selling prices. As such, the AM price for 5/1 should be smaller than selling prices for 4/30 and 5/2, while the PM price should be larger than the selling prices for 4/30 and 5/2. Additionally, the PM price should be slightly larger than the AM price.

We can take another look at the prices for the week with the missing prices as a table:

```
turnips %>%
  filter(date >= "2020-04-26" & date <= "2020-05-02")
```

```
## # A tibble: 7 x 5
##   date       buy_price sell_price_am sell_price_pm weekday
##   <date>         <dbl>         <dbl>         <dbl> <fct>
## 1 2020-04-26       110            NA            NA Sunday
## 2 2020-04-27        NA            96            92 Monday
## 3 2020-04-28        NA           138           220 Tuesday
## 4 2020-04-29        NA           385           201 Wednesday
## 5 2020-04-30        NA           123            90 Thursday
## 6 2020-05-01        NA            NA            NA Friday
## 7 2020-05-02        NA            58            52 Saturday
```

From these observations, it seems reasonable to use 50 bells for the AM selling price, and 100 bells for the PM selling price. We can now impute these values into the data set:

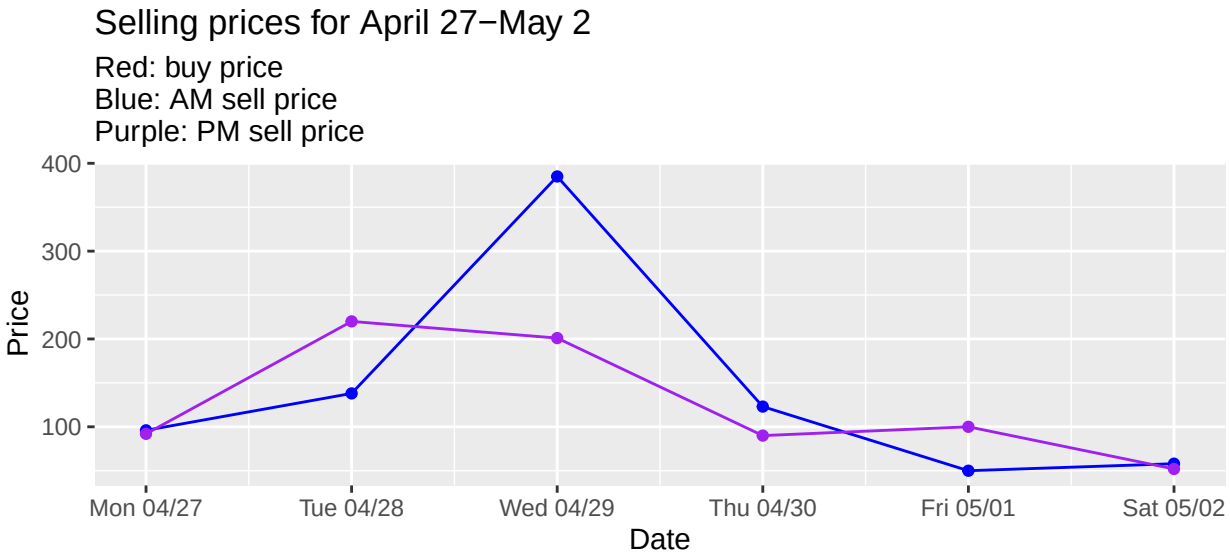
```
turnips = turnips %>%
  mutate(sell_price_am = if_else(date == "2020-05-01", 50, sell_price_am)) %>%
  mutate(sell_price_pm = if_else(date == "2020-05-01", 100, sell_price_pm))
turnips
```

```
## # A tibble: 84 x 5
##    date       buy_price sell_price_am sell_price_pm weekday
##    <date>         <dbl>         <dbl>         <dbl> <fct>
##  1 2020-04-26       110            NA            NA Sunday
##  2 2020-04-27        NA            96            92 Monday
##  3 2020-04-28        NA           138           220 Tuesday
##  4 2020-04-29        NA           385           201 Wednesday
##  5 2020-04-30        NA           123            90 Thursday
##  6 2020-05-01        NA            50           100 Friday
##  7 2020-05-02        NA            58            52 Saturday
##  8 2020-05-03        98            NA            NA Sunday
##  9 2020-05-04        NA            84            80 Monday
## 10 2020-05-05        NA            76            72 Tuesday
## # ... with 74 more rows
```

We can now recreate the plot for this week with the imputed values:

```
turnips %>%
  filter(date <= "2020-05-02" & date > "2020-04-26") %>%
  ggplot(aes(x = date))+
  geom_line(aes(y = sell_price_am), na.rm = T, color = 'blue')+
  geom_point(aes(y = sell_price_am), na.rm = T, color = 'blue')+
  geom_line(aes(y = sell_price_pm), na.rm = T, color = 'purple')+
  geom_point(aes(y = sell_price_pm), na.rm = T, color = 'purple')+
```

```
  scale_x_date(date_labels = "%a %m/%d", date_breaks = "day")+
  labs(x = "Date", y = "Price", title = "Selling prices for April 27-May 2",
       subtitle = "Red: buy price\nBlue: AM sell price\nPurple: PM sell price")
```



While other methods, such as imputing the mean for other weeks, may have been useful, small variations in the other weeks may have made the prices less consistent. For instance, using the mean Friday selling price may cause the pattern to break away - we want to ensure that Friday's AM selling price is lower than Thursday and Saturday's AM selling prices and that Friday's PM selling price is higher than Thursday and Saturday's. While 50 and 100 are probably not the true/exact prices, they provide estimates for what the selling price is likely to be near.

6. Probability of profitability

The primary goal of the Stalk market is to make a profit on our initial investment. Some weeks provide the opportunity for very large returns, with selling prices peaking near or even above 500 bells. However, some weeks provide no opportunity for gains, as the selling price only drops each day. How can we maximize our profits? One way to begin is to look at the average selling price each day in the morning and afternoon:

```
turnips %>%
  group_by(weekday) %>%
  filter(weekday != "Sunday") %>%
  summarise(mean_sell_am = mean(sell_price_am, na.rm = T),
            mean_sell_pm = mean(sell_price_pm, na.rm = T)) %>%
  arrange(desc(mean_sell_am))
```

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

```
## # A tibble: 6 x 3
##   weekday   mean_sell_am mean_sell_pm
##   <fct>            <dbl>        <dbl>
## 1 Wednesday        218.         119.
## 2 Tuesday          108.         126.
## 3 Thursday          96.8         72.9
## 4 Monday            88.9         92.6
## 5 Friday            66.1         67.3
## 6 Saturday          59.8         63.5
```

Based solely on averages, Wednesday provides the best morning selling price and Tuesday provides the best afternoon selling price. However, these averages are mostly influenced by weeks where the price spikes for these two periods of time, then goes back down. We cannot rely on this spike occurring every week to make a profit, as there may be some weeks where other days have a higher price, or where the selling price stays lower than the buying price for the whole week.

In order to more easily analyze each week individually, and compare weeks to one another, we can add a new column for the week number. The `epiweek` function from the `lubridate` package allows us to quickly add such a column.

```
turnips = turnips %>%
  mutate(week_num = as.integer(epiweek(date)))
turnips
```

```
## # A tibble: 84 x 6
##    date       buy_price sell_price_am sell_price_pm weekday   week_num
##    <date>         <dbl>         <dbl>         <dbl> <fct>        <int>
##  1 2020-04-26       110            NA            NA Sunday          18
##  2 2020-04-27        NA            96            92 Monday          18
##  3 2020-04-28        NA           138           220 Tuesday         18
##  4 2020-04-29        NA           385           201 Wednesday       18
##  5 2020-04-30        NA           123            90 Thursday        18
##  6 2020-05-01        NA            50           100 Friday          18
##  7 2020-05-02        NA            58            52 Saturday        18
##  8 2020-05-03        98            NA            NA Sunday          19
##  9 2020-05-04        NA            84            80 Monday          19
## 10 2020-05-05        NA            76            72 Tuesday         19
## # ... with 74 more rows
```

We can calculate how many of the selling prices in a given week were larger than the buying price. Using the `fill` function, we can impute the `NA` values in the `buy_price` column with that week's buy price to make this calculation easier.

```
turnips %>%
  group_by(week_num) %>%
  fill(buy_price) %>%
  summarise(profitable_times = sum(sell_price_am > buy_price, na.rm = T)+
              sum(sell_price_pm > buy_price, na.rm = T),
            profitable_pct = profitable_times/12) %>%
  print() %>%
  ungroup() %>%
  summarise(total_profitable_times = sum(profitable_times),
    overall_profit_pct = mean(profitable_pct))
```

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

```
## # A tibble: 12 x 3
##    week_num profitable_times profitable_pct
##       <int>            <int>          <dbl>
##  1       18                5          0.417
##  2       19                0          0
##  3       20                7          0.583
##  4       21                0          0
##  5       22                5          0.417
##  6       23                6          0.5
##  7       24                5          0.417
##  8       25                0          0
##  9       26                5          0.417
## 10       27                5          0.417
## 11       28                7          0.583
## 12       29                4          0.333
```

```
## # A tibble: 1 x 2
##   total_profitable_times overall_profit_pct
##                    <int>              <dbl>
## 1                     49              0.340
```

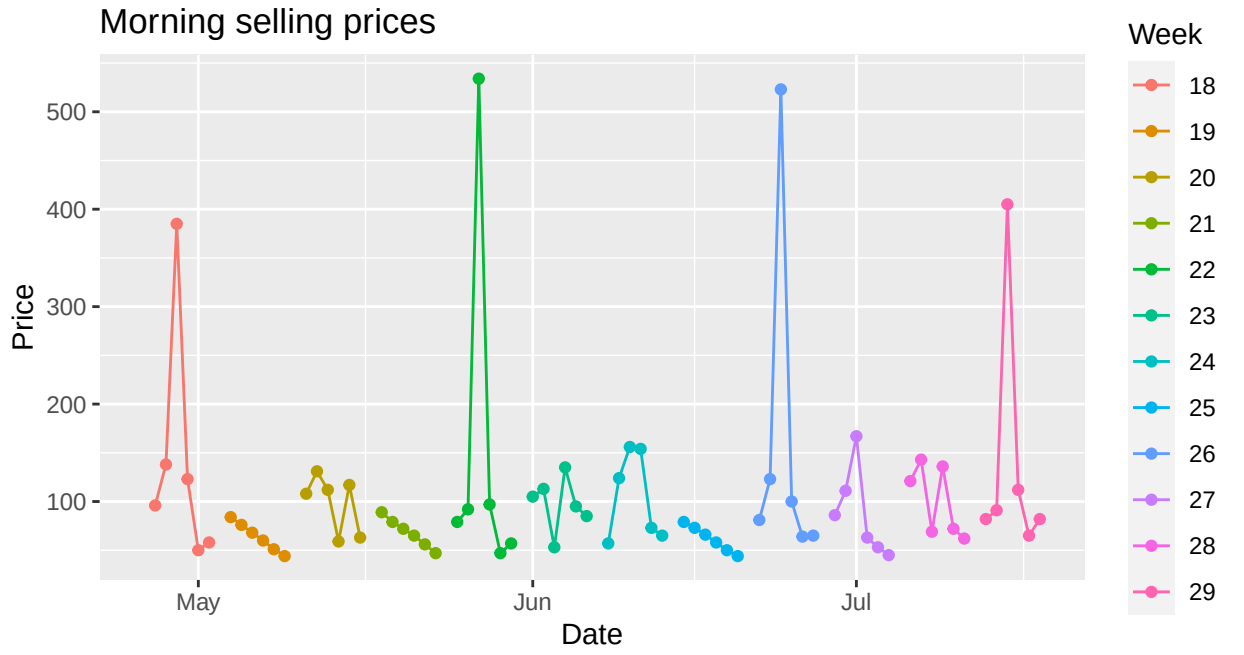
The first table shows each week's number of time periods (6 day period of selling prices) that are profitable, alongside the week's corresponding percentage if profitable time period. Each day, Monday to Saturday, has two selling prices for a total of twelve selling prices. The second table takes the mean of the percent of profitable days, giving us the percentage of profitable days in the entire data set. With 12 weeks and 12 time periods per week, we have a total of 144 time periods.

What this tells us is throughout this 12 week span of prices, only about 34% of the selling prices were above that week's buying price. While some weeks may have selling prices that are higher than the buying price for over 50% of the time periods, other weeks provide no opportunity to make a profit. As such, there is about a 34% chance of being able to make a profit each week in the long run. However, this does not include *how much* profit is made each week, nor if these profits make up for previous losses. Rather, we wanted to address *if* profits would be attainable throughout the week.

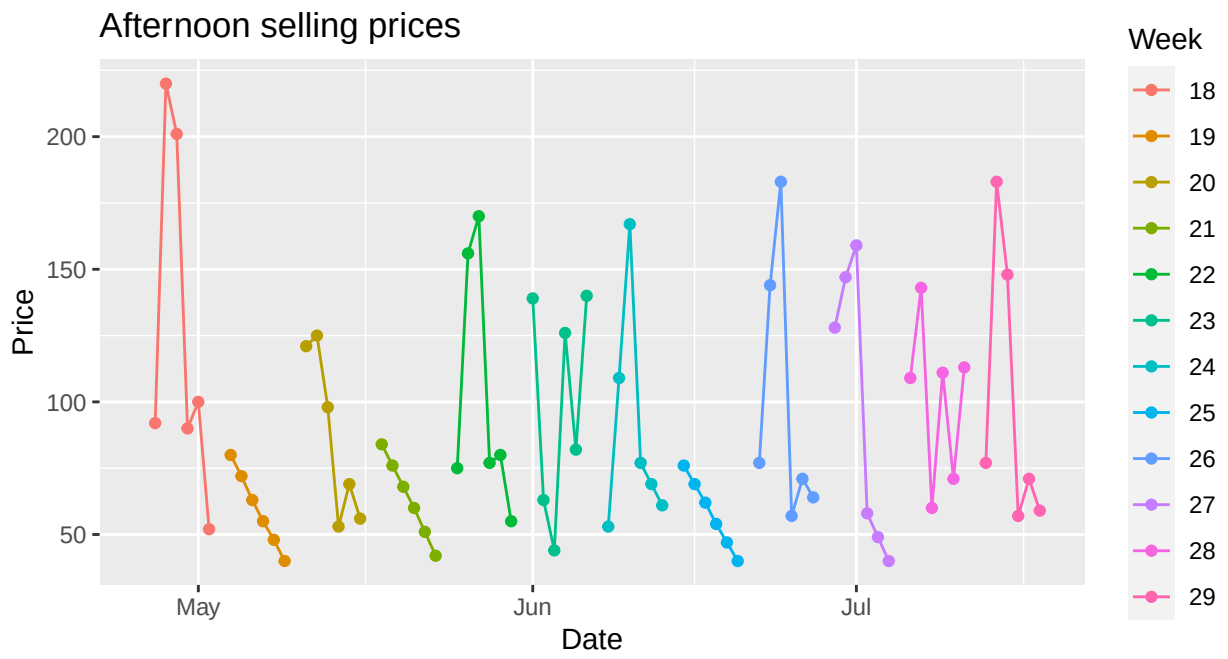
7. Selling price patterns

With the goal of ensuring a profit, or at least minimizing losses, at hand, this brings up a new question. We've seen a few weekly patterns by plotting weekly selling prices - for instance, when the prices spike on Wednesday, or when they decrease throughout the week. How many of these unique patterns are there? To figure this out, we can start by taking another look at the weekly prices:

```
ggplot(turnips, aes(x = date, y = sell_price_am))+
  geom_line(na.rm = T, aes(color = as.factor(week_num)))+
  geom_point(na.rm = T, aes(color = as.factor(week_num)))+
  labs(x = "Date", y = "Price",
       title = "Morning selling prices", color = "Week")
```

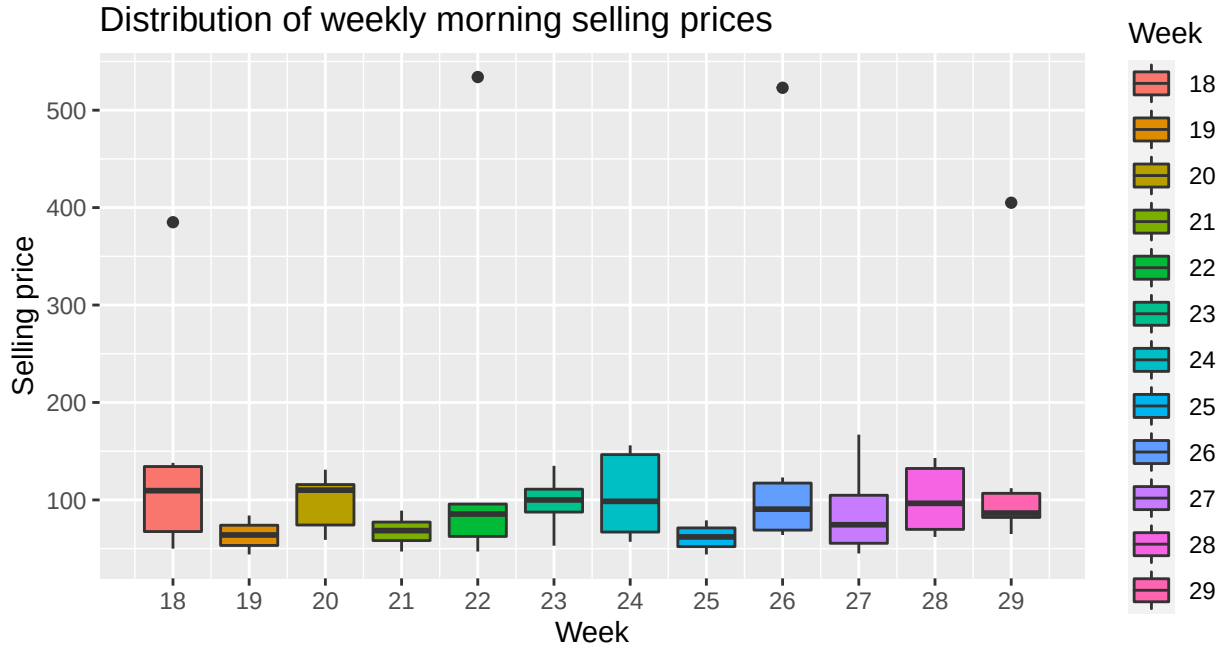



```
ggplot(turnips, aes(x = date, y = sell_price_pm))+
  geom_line(na.rm = T, aes(color = as.factor(week_num)))+
  geom_point(na.rm = T, aes(color = as.factor(week_num)))+
  labs(x = "Date", y = "Price",
       title = "Afternoon selling prices", color = "Week")
```

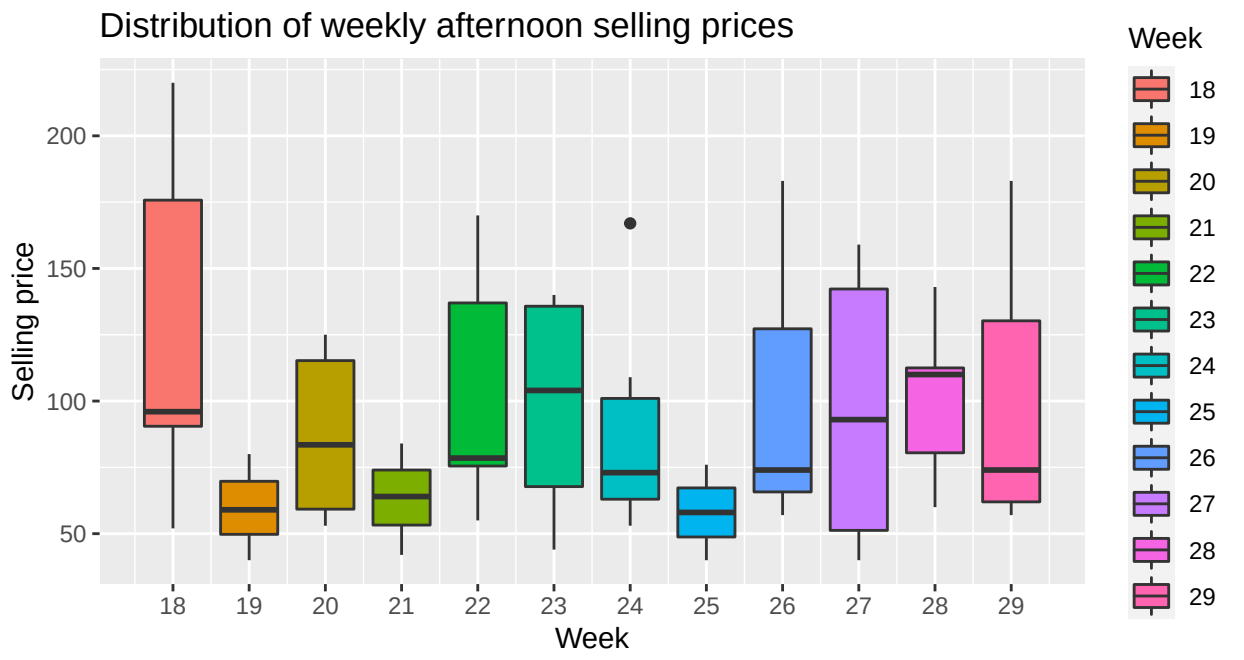


Additionally, we can create boxplots for each week's selling prices to see how the prices are distributed within each week.

```
ggplot(turnips)+
  geom_boxplot(aes(x = week_num, y = sell_price_am,
                   group = week_num, fill = as.factor(week_num)), na.rm = T)+
  scale_x_continuous(breaks = seq(17, 30))+
  labs(title = "Distribution of weekly morning selling prices",
       x = "Week", y = "Selling price", fill = "Week")
```

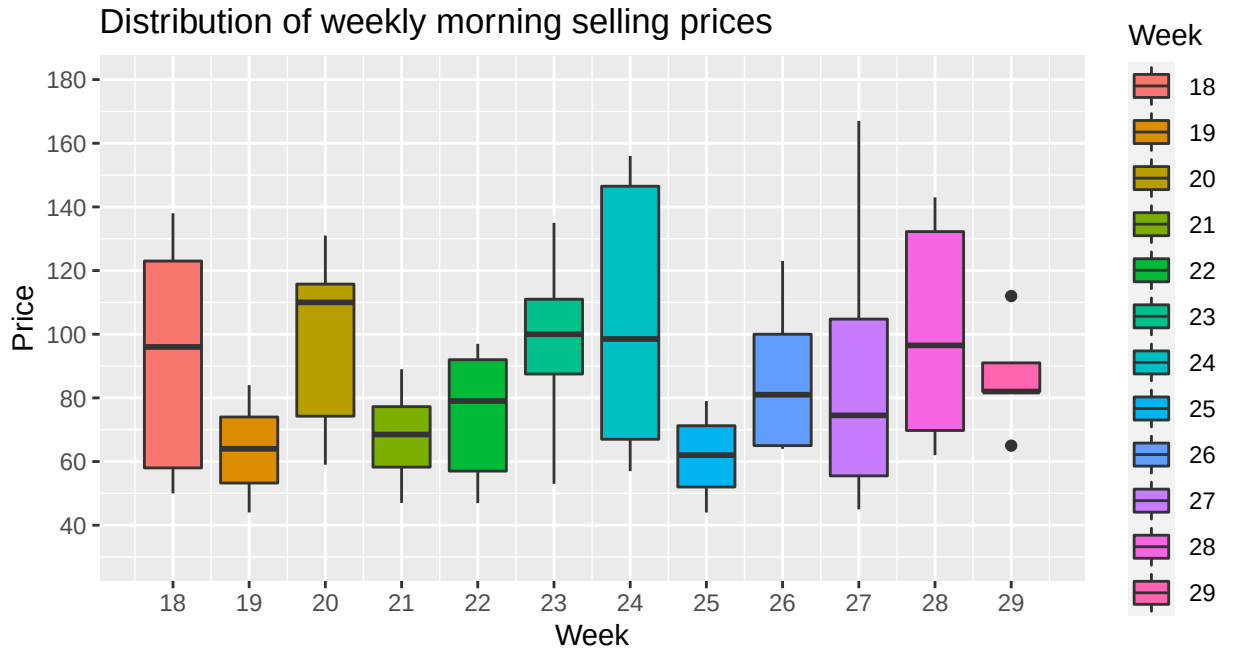


```
ggplot(turnips)+
  geom_boxplot(aes(x = week_num, y = sell_price_pm,
                   group = week_num, fill = as.factor(week_num)), na.rm = T)+
  scale_x_continuous(breaks = seq(17, 30))+
  labs(title = "Distribution of weekly afternoon selling prices",
       x = "Week", y = "Selling price", fill = "Week")
```



The influence of the outliers makes the boxplot of morning selling prices more difficult to read, so let's zoom in:

```
ggplot(turnips)+
  geom_boxplot(aes(x = week_num, y = sell_price_am,
                   group = week_num, fill = as.factor(week_num)), na.rm = T)+
  scale_x_continuous(breaks = seq(17, 30))+
  scale_y_continuous(breaks = seq(40, 200, 20), limits = c(30,180))+
  labs(title = "Distribution of weekly morning selling prices",
       x = "Week", y = "Price", fill = "Week")
```



Now that we have a better look at the boxes, we can see how a few weeks are nearly identical to one another. Weeks 19, 21, and 25 all look very similar. Their morning selling prices all have medians between 60 and 70 and have a very symmetric distribution. Referring back to the scatterplot, we can see that these boxes correspond to the weeks that experience a steady decline throughout the week. Weeks 18, 22, 26, and 29 all have right skewed distributions, along with a single outlier in the 350-550 range. Weeks 24 and 27 seem to follow this trend as well, but with a much smaller peak of around 150 bells. The fourth and final weekly pattern is observed in weeks 20, 23, and 28. The pattern is much more noticeable on the scatterplot than the boxplot, but these weeks do have quite large interquartile ranges and/or long whiskers. The prices fluctuate throughout the week, with an occasional peak around 100-150 bells.

To summarize, the following four patterns are observed in the following weeks:

- Linear decrease in weeks 19, 21, and 25
- Large spike in weeks 18, 22, 26, and 29
- Small spike in weeks 24 and 27
- Fluctuating in weeks 20, 23, and 28

We can create a new column to add these descriptors to each week.

```
turnips = turnips %>%
  mutate(week_pattern = if_else(week_num %in% c(18, 22, 26, 29), "largespike",
                                if_else(week_num %in% c(19, 21, 25), "decreasing",
                                if_else(week_num %in% c(24, 27), "smallspike",
                                if_else(week_num %in% c(20, 23, 28), "fluctuating", "")))))
turnips %>%
  count(week_pattern) # verify mutation worked as intended
```

```
## # A tibble: 4 x 2
##   week_pattern     n
##   <chr>        <int>
## 1 decreasing      21
## 2 fluctuating     21
```

```
## 3 largespike      28
## 4 smallspike      14
```

Everything seems to be in order - each week pattern occurs in our data set as often as we expect, i.e. decreasing weeks appear 21 times (3 weeks - weeks 19, 21, and 25). While we may or may not end up needing this new column, it can still be useful to have the weeks labeled so we know which pattern each week follows.

8. Predicting selling price patterns

Now that we've discovered the underlying patterns that can occur, can we predict, given the current week's pattern, what the next week's pattern will be? We can recreate our previous plots using our new `week_pattern` variable to color the weeks. Plotting the morning selling prices with this coloring allows us to easily see how the patterns change from week to week. Plotting the afternoon prices isn't necessary for this, as we are only interested in the pattern change week to week, not the prices.

```
ggplot(turnips, aes(date, sell_price_am))+
  geom_line(aes(color = week_pattern), na.rm = T)+
  geom_point(aes(color = week_pattern), na.rm = T)+
  labs(x = "Date", y = "Selling price",
       title = "Morning selling prices",color = "Pattern")
```

The first thing to note is that no two consecutive weeks have the same selling price pattern. While we cannot say that the probability of the same pattern occurring two weeks in a row is zero, it certainly seems to be highly unlikely. As such, the probabilities we calculate will be from our data, so they may not be very representative of the true probabilities. For the three decreasing weeks, two large spikes and one fluctuating pattern occurred afterwards. For the large spike weeks (excluding the last week), the three other patterns occurred once each, so we can assume that a large spike one week yields an approximately equal probability for one of the other three patterns to occur the next week. The same phenomenon occurs for weeks with a fluctuating pattern. The small spike only occurred twice, and these were followed by a decreasing pattern one week and a fluctuating pattern the other.

To summarize, we'll group these probabilities into a table:

Next Week	This Week			
	Large spike	Small spike	Decreasing	Fluctuating
Large spike	~0%	~0%	~66%	~33%
Small spike	~33%	~0%	~0%	~33%
Decreasing	~33%	~50%	~0%	~33%
Fluctuating	~33%	~50%	~33%	~0%

Approximations are used since we cannot be certain of the exact underlying probabilities that the game uses. However, these probabilities can still give us an idea of which pattern to expect for the following week based on what this week's pattern is. For instance, if we get a large spike this week, we would be more inclined to expect the following week to be a decreasing or fluctuating pattern, and be almost certain it will *not* be another large spike pattern.

9. Pricing of turnips on certain days

We know that the selling prices varies from day to day, with some days seeking consistent prices based on that week's particular instance. For instance, Tuesday afternoon and Wednesday morning prices are always very high on a week with a large spike pattern. However, we want to assess the significance of these differences.

Before diving too deep into statistical tests, we can do some basic data exploration to investigate the average selling prices for each day.

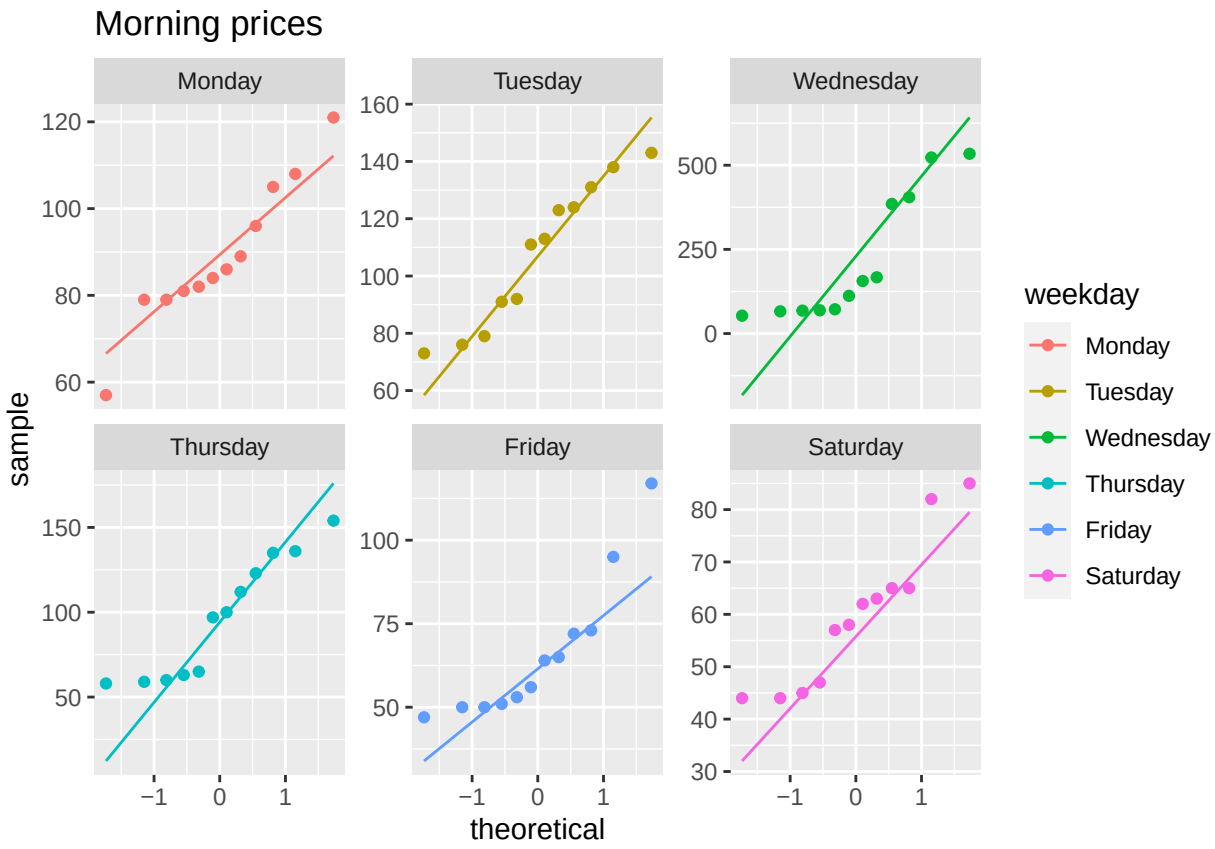
```
turnips %>%
  filter(weekday != "Sunday") %>%
  group_by(weekday) %>%
  summarise(am_avg = mean(sell_price_am, na.rm = T),
            pm_avg = mean(sell_price_pm, na.rm = T))
```

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

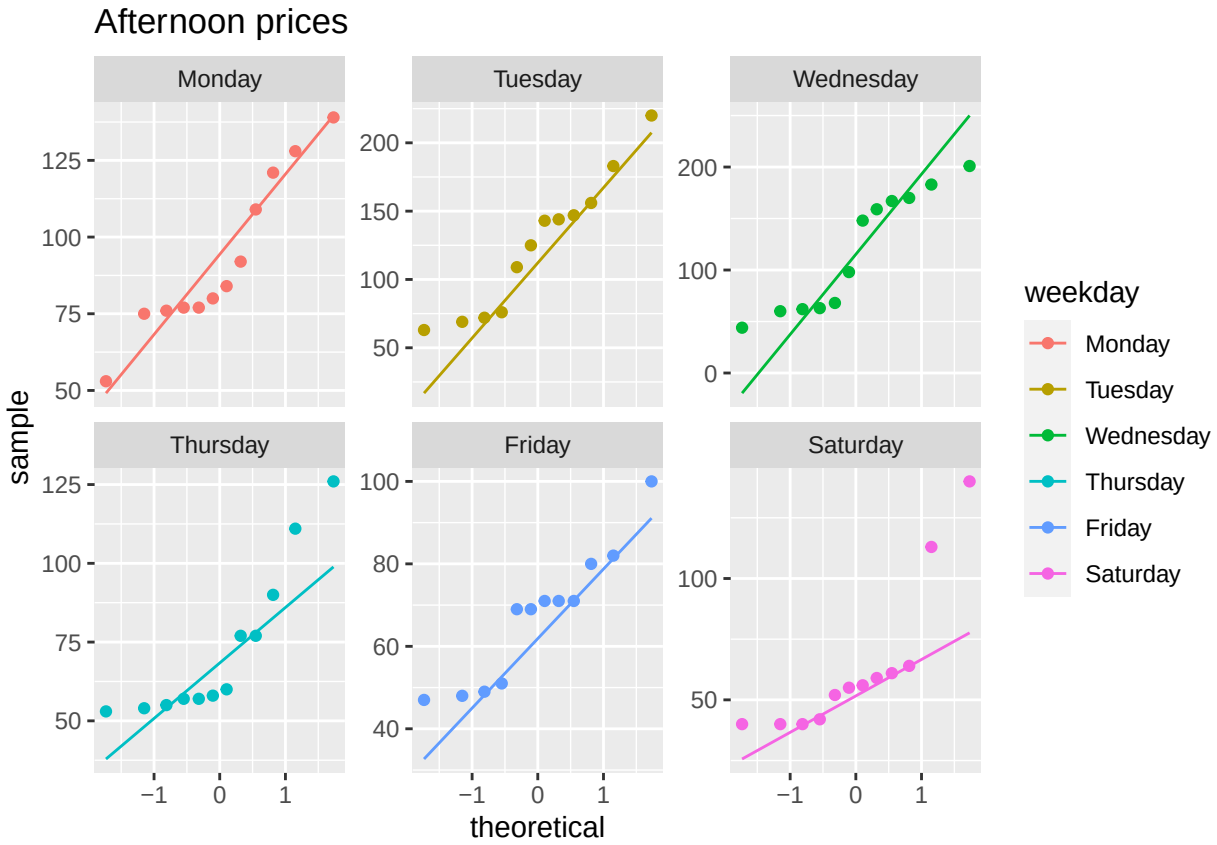
```
## # A tibble: 6 x 3
##   weekday   am_avg pm_avg
##   <fct>      <dbl>  <dbl>
## 1 Monday      88.9   92.6
## 2 Tuesday    108.   126.
## 3 Wednesday  218.   119.
## 4 Thursday    96.8   72.9
## 5 Friday      66.1   67.3
## 6 Saturday    59.8   63.5
```

We can see that Wednesday and Thursday have the highest average selling prices while Friday and Saturday have the lowest.

```
turnips %>%
  filter(weekday != "Sunday") %>%
  group_by(weekday) %>%
  ggplot(aes(sample = sell_price_am, color = weekday))+
  stat_qq()+
  stat_qq_line()+
  facet_wrap(.~weekday, scales = "free_y")+
  labs(title = "Morning prices")
```




```
turnips %>%
  filter(weekday != "Sunday") %>%
  group_by(weekday) %>%
  ggplot(aes(sample = sell_price_pm, color = weekday))+
  stat_qq()+
  stat_qq_line()+
  facet_wrap(.~weekday, scales = "free_y")+
  labs(title = "Afternoon prices")
```



It can be a bit difficult to tell how much these samples deviate from a normal distribution due to the small sample size (each weekday only has 12 data points). For instance, Saturday appears to be mostly normal, but has very large outliers on the right tail, while the other weeks have most of their points far from the line.

Now that we have an idea of how different these averages are, just how significant are these differences? Due to strong suggestions of non-normality, as first found in question 4, and now here with the QQ plots of selling prices, using parametric tests may give us misleading results.

While a test such as Tukey's HSD (honest significant difference) may provide some insight into the differences, the results may not be statistically accurate. As such, it would be preferable to use a nonparametric multiple comparisons test, such as the Kruskal-Wallis rank sum test [2]. Further details, in particular the test statistic, can be found in [2]. To very briefly summarize, this test is a rank test that compares population medians (with ranking). Using the median and ranks instead of the raw data helps accounts for the lack of normality in the data. This gives us the following hypotheses:

$$H_0 : m_i = m_j \ \forall i, j, \ i \neq j; \ 1 \leq i, j \leq 6$$
$$H_a : m_i \neq m_j \text{ for any } i, j; i \neq j,$$

where m_i and m_j are the medians of weeks i and j, respectively. So, H_0 dictates that every week's median is equal to every other week's median. As long as the medians of at least one pair are different, we can reject H_0 in favor of H_a. We perform the Krukal-Wallis test for the morning and afternoon selling prices:

```
kruskal.test(sell_price_am~weekday, turnips)
```

```
##
```

```
##  Kruskal-Wallis rank sum test
##
## data:  sell_price_am by weekday
## Kruskal-Wallis chi-squared = 27.797, df = 5, p-value = 3.988e-05
```

```
kruskal.test(sell_price_pm~weekday, turnips)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  sell_price_pm by weekday
## Kruskal-Wallis chi-squared = 22.742, df = 5, p-value = 0.0003781
```

From our previous findings regarding the weekly patterns, this does not come as a surprise that at least one group's mean is significantly different from another. For instance, Wednesday's selling prices likely have a much higher average than most of the other days due to the large spike pattern inflating Wednesday's average. However, this result only tells us that *at least one* pairing has significantly different averages. Tukey's HSD can provide us with a little more insight for each pairing. While we should exercise caution in drawing our conclusions from this test due to the possible non-normality present in the data, it can still give us an idea of how different each pair of days is.

```
TukeyHSD(aov(sell_price_am~weekday, turnips))
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = sell_price_am ~ weekday, data = turnips)
##
## $weekday
##                           diff        lwr       upr     p adj
## Tuesday-Monday       18.916667  -76.69935 114.53269 0.9919847
## Wednesday-Monday    128.583333   32.96731 224.19935 0.0025768
## Thursday-Monday       7.916667  -87.69935 103.53269 0.9998789
## Friday-Monday       -22.833333 -118.44935  72.78269 0.9812036
## Saturday-Monday     -29.166667 -124.78269  66.44935 0.9463155
## Wednesday-Tuesday   109.666667   14.05065 205.28269 0.0154331
## Thursday-Tuesday    -11.000000 -106.61602  84.61602 0.9993927
## Friday-Tuesday      -41.750000 -137.36602  53.86602 0.7939115
## Saturday-Tuesday    -48.083333 -143.69935  47.53269 0.6804988
## Thursday-Wednesday -120.666667 -216.28269 -25.05065 0.0055926
## Friday-Wednesday   -151.416667 -247.03269 -55.80065 0.0002323
## Saturday-Wednesday -157.750000 -253.36602 -62.13398 0.0001148
## Friday-Thursday     -30.750000 -126.36602  64.86602 0.9334407
## Saturday-Thursday   -37.083333 -132.69935  58.53269 0.8635273
## Saturday-Friday      -6.333333 -101.94935  89.28269 0.9999598
```

```
TukeyHSD(aov(sell_price_pm~weekday, turnips))
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = sell_price_pm ~ weekday, data = turnips)
```

```
##
## $weekday
##                           diff        lwr        upr     p adj
## Tuesday-Monday      33.000000  -11.53376  77.533762 0.2633335
## Wednesday-Monday    26.000000  -18.53376  70.533762 0.5279542
## Thursday-Monday    -19.666667  -64.20043  24.867096 0.7860648
## Friday-Monday      -25.250000  -69.78376  19.283762 0.5599176
## Saturday-Monday    -29.083333  -73.61710  15.450429 0.4013374
## Wednesday-Tuesday   -7.000000  -51.53376  37.533762 0.9972720
## Thursday-Tuesday   -52.666667  -97.20043  -8.132904 0.0113585
## Friday-Tuesday     -58.250000 -102.78376 -13.716238 0.0036516
## Saturday-Tuesday   -62.083333 -106.61710 -17.549571 0.0015992
## Thursday-Wednesday -45.666667  -90.20043  -1.132904 0.0412334
## Friday-Wednesday   -51.250000  -95.78376  -6.716238 0.0149347
## Saturday-Wednesday -55.083333  -99.61710 -10.549571 0.0070241
## Friday-Thursday     -5.583333  -50.11710  38.950429 0.9990783
## Saturday-Thursday   -9.416667  -53.95043  35.117096 0.9891270
## Saturday-Friday     -3.833333  -48.36710  40.700429 0.9998533
```

The results of these tests go to show just how much Wednesday's spikes affects its average compared to the other days, particularly for the morning selling prices. In fact, Wednesday's pairings for the morning selling prices are the only ones that are significant at a threshold of $\alpha = 0.05$. For instance, the p-value for Saturday-Wednesday (AM) is only 0.0001. However, the results are bit more varied for the afternoon selling prices. There are six significantly different pairings, three of which are with Tuesday, the other three with Wednesday. The other pairings are insignificant, with some p-values very close to 1.

So, to summarize our findings:

- Wednesday morning is larger than every other morning ($p<0.05$)
- Tuesday and Wednesday afternoons are larger than Thursday, Friday, and Saturday afternoons

As such, the varying patterns that occur cause certain days to have prices that are not too different from one another. We would not expect pairings like Saturday and Monday to provide us with anything remarkable, as previous findings have shown how these days' selling prices don't stand out much, regardless of the pattern. On the other hand, Wednesday's afternoon selling price has the occasional spike of 300-500 bells, which greatly influences its average.

10. Predicting future selling prices

Now, we want to see if we can predict future selling prices. We'll have two scenarios with different selling prices. In particular, given the Sunday buying price and selling prices for Monday, Tuesday, and Wednesday, we want to try and predict what the selling prices for Thursday, Friday, and Saturday would be.

Scenario 1: Buying price is 93 bells on Sunday. Selling prices are 140 (AM) and 127 (PM) on Monday, 183/212 on Tuesday, 158/83 on Wednesday

For consistency and convenience, we can add these values to our existing data set. Since the week's pattern is not known for certain, we will leave `week_pattern` as unknown.

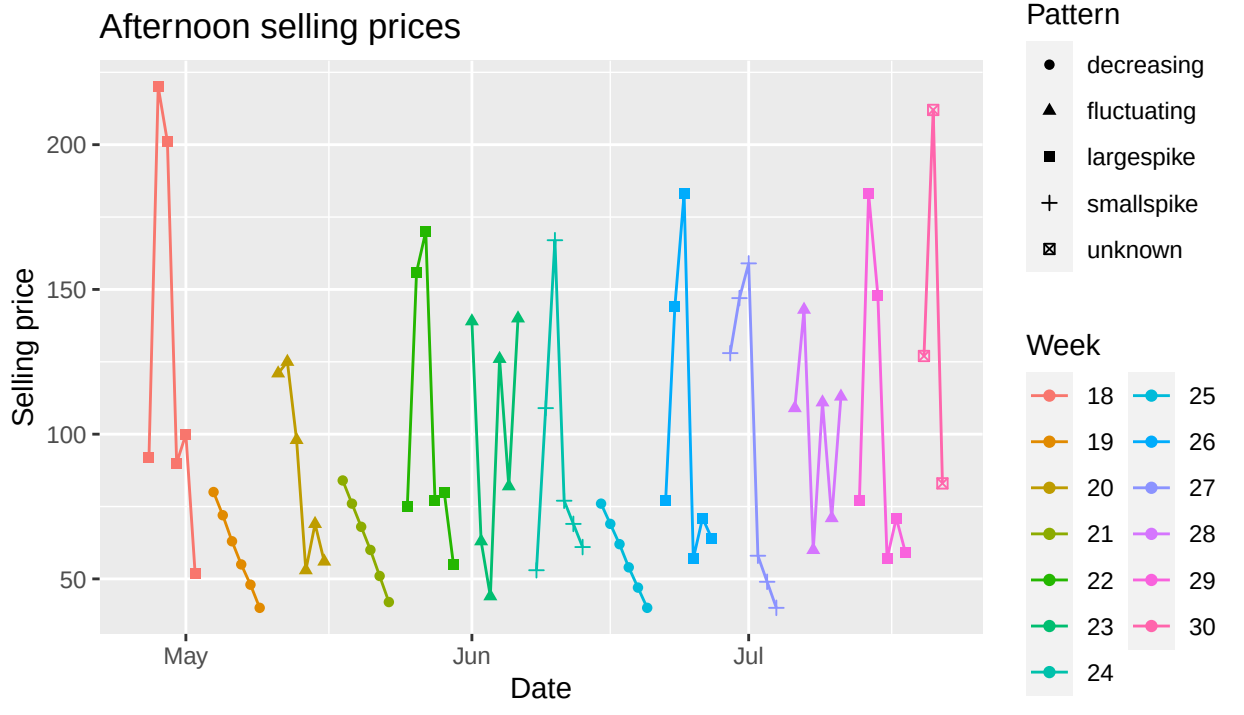
```
turnips_s1 = turnips %>%
  add_row(date = as.Date(c("2020-07-19", "2020-07-20",
                           "2020-07-21", "2020-07-22")),
          buy_price = c(93, rep(NA, 3)),
          sell_price_am = c(NA, 140, 183, 158),
          sell_price_pm = c(NA, 127, 212, 83),
          weekday = weekdays(date), week_num = as.integer(epiweek(date)),
          week_pattern = "unknown")

tail(turnips_s1, 5)
```

```
## # A tibble: 5 x 7
##   date       buy_price sell_price_am sell_price_pm weekday week_num week_pattern
##   <date>         <dbl>         <dbl>         <dbl> <chr>      <int> <chr>
## 1 2020-07-18        NA            82            59 Saturd~       29 largespike
## 2 2020-07-19        93            NA            NA Sunday        30 unknown
## 3 2020-07-20        NA           140           127 Monday        30 unknown
## 4 2020-07-21        NA           183           212 Tuesday       30 unknown
## 5 2020-07-22        NA           158            83 Wednes~       30 unknown
```

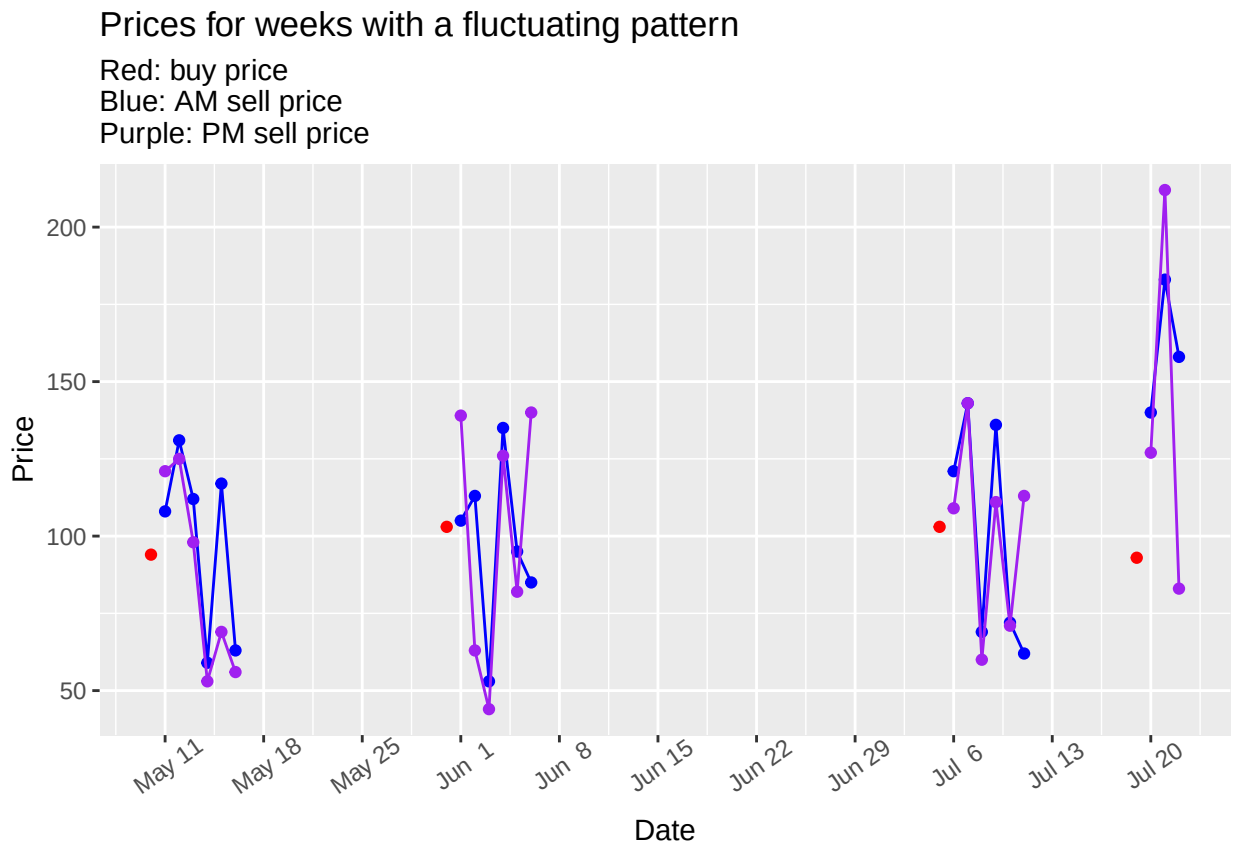
```
ggplot(turnips_s1, aes(date, sell_price_am, shape = week_pattern,
                       color = as.factor(week_num)))+
  geom_point(na.rm = T)+
  geom_line(na.rm = T)+
  labs(x = "Date", y = "Selling price", color = "Week", shape = "Pattern",
       title = "Morning selling prices")+
  guides(col = guide_legend(nrow = 7))
```


```
ggplot(turnips_s1, aes(date, sell_price_pm, shape = week_pattern,
                       color = as.factor(week_num)))+
  geom_point(na.rm = T)+
  geom_line(na.rm = T)+
  labs(x = "Date", y = "Selling price", color = "Week", shape = "Pattern",
       title = "Afternoon selling prices")+
  guides(col = guide_legend(nrow = 7))
```



Our new fictional "week 30" seems to bear the most resemblance to week 28, which has a fluctuating pattern. However, it would be less than optimal to limit ourselves to compare it to only one other week. We can plot the prices of the fluctuating weeks along with our scenario's prices (we'll keep calling it week 30) to compare:

```
turnips_s1 %>%
         filter(week_pattern == "fluctuating" | week_num == 30) %>%
  ggplot(aes(x = date))+
  geom_point(aes(y = buy_price), color = 'red', na.rm = T)+
  geom_line(aes(y = sell_price_am, group = week_num), color = 'blue', na.rm = T)+
  geom_point(aes(y = sell_price_am, group = week_num), color = 'blue', na.rm = T)+
  geom_line(aes(y = sell_price_pm, group = week_num), color = 'purple', na.rm = T)+
  geom_point(aes(y = sell_price_pm, group = week_num), color = 'purple', na.rm = T)+
  labs(title = "Prices for weeks with a fluctuating pattern",
       subtitle = "Red: buy price\nBlue: AM sell price\nPurple: PM sell price",
       x = "Date", y = "Price")+
  scale_x_date(breaks = "1 week", date_labels = "%b %e")+
  theme(axis.text.x = element_text(angle = 35, vjust = 0.8))
```



The maximum and minimum values for week 28 occur on Tuesday and Wednesday respectively. Additionally, the prices on Monday, Wednesday, and Friday afternoons in week 28 are almost identical. Week 30 seems to resemble a scaled up week 28, which may be the best in assisting with our prediction. However, we'll still calculate some summary statistics for all three of these weeks to try and get some more insight.

```
turnips_s1 %>%
  filter(week_pattern == "fluctuating" | week_num == 30) %>%
```

```
  group_by(week_num) %>%
  summarise(mean_am = mean(sell_price_am, na.rm = T),
            mean_pm = mean(sell_price_pm, na.rm = T),
            range_am = range(sell_price_am, na.rm = T),
            range_pm = range(sell_price_pm, na.rm = T))
```

```
## `summarise()` regrouping output by 'week_num' (override with `.groups` argument)
```

```
## # A tibble: 8 x 5
## # Groups:   week_num [4]
##   week_num mean_am mean_pm range_am range_pm
##      <int>   <dbl>   <dbl>    <dbl>    <dbl>
## 1       20    98.3     87        59       53
## 2       20    98.3     87       131      125
## 3       23    97.7     99        53       44
## 4       23    97.7     99       135      140
## 5       28   100.     101.       62       60
## 6       28   100.     101.      143      143
## 7       30   160.     141.      140       83
## 8       30   160.     141.      183      212
```

We can look at how much the prices for week 30 are scaled up compared to week 28:

```
week28 = turnips_s1 %>%
  filter(week_num == 28 & !is.na(sell_price_am)) %>%
  select(sell_price_am, sell_price_pm)

week30 = turnips_s1 %>%
  filter(week_num == 30 & !is.na(sell_price_am)) %>%
  select(sell_price_am, sell_price_pm)

# only compare Monday Tuesday and Wednesday
week30/week28[1:3,]
```

```
##   sell_price_am sell_price_pm
## 1      1.157025      1.165138
## 2      1.279720      1.482517
## 3      2.289855      1.383333
```

We can see that the AM selling prices are anywhere from 1.15 to 2.28 times larger for week 30, while the PM prices range from being 1.16 to 1.38 times larger. We can use this information to help us scale up week 28's values and fill them into week 30's prices. We'll extract week 28's prices and impute them into week 30's selling price rows.

```
week28_am = turnips_s1 %>%
  filter(week_num == 28 & !is.na(sell_price_am)) %>%
  select(sell_price_am) %>%
  pull(sell_price_am)

week28_pm = turnips_s1 %>%
  filter(week_num == 28 & !is.na(sell_price_pm)) %>%
```

```
  select(sell_price_pm) %>%
  pull(sell_price_pm)

week28_am
```

```
## [1] 121 143  69 136  72  62
```

```
week28_pm
```

```
## [1] 109 143  60 111  71 113
```

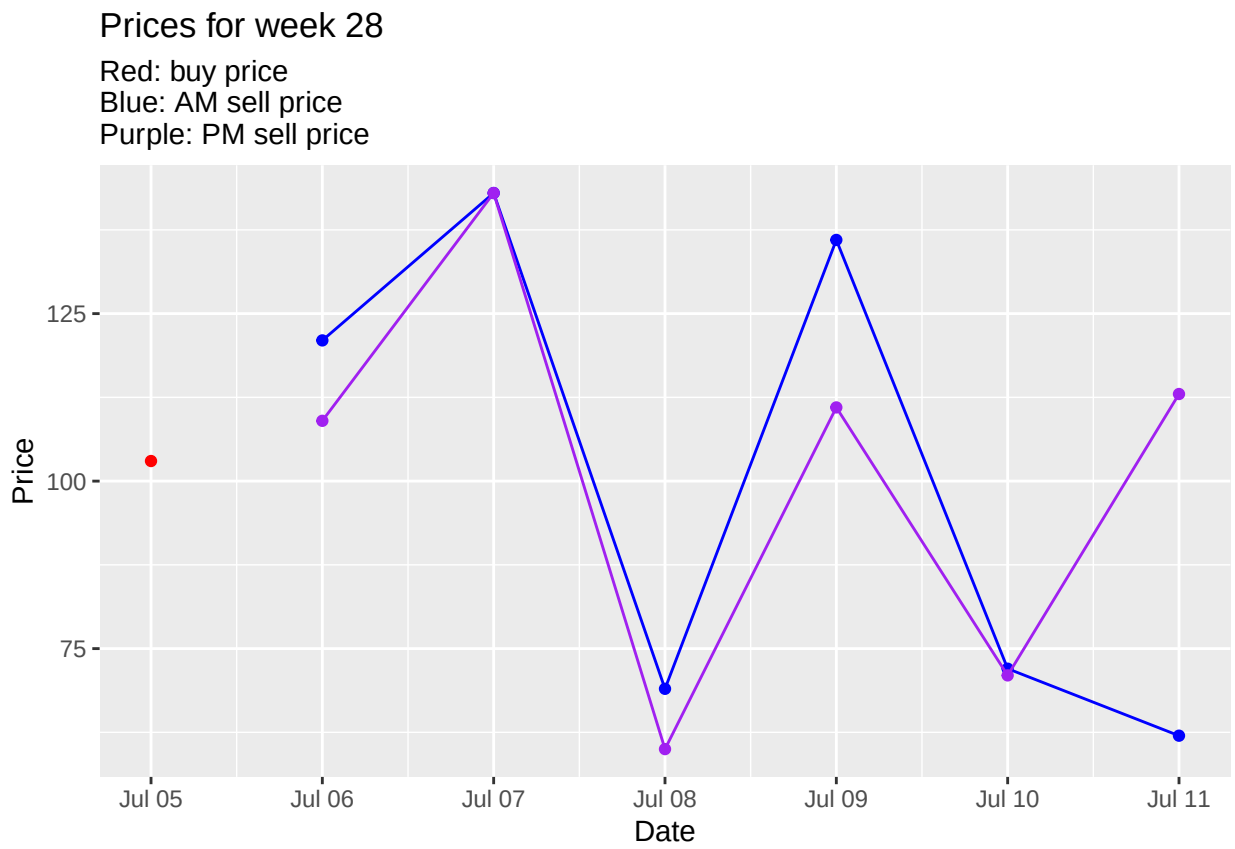
Now that we have each day's prices, we can scale up these values and impute them into the data frame for week 30's prices. Since we already have values for Monday, Tuesday, and Wednesday, we'll only modify the selling prices for Thursday, Friday, and Saturday.

```
turnips_s1 %>%
  filter(week_num == 30) %>%
  add_row(date = as.Date(c("2020-07-23", "2020-07-24",
                           "2020-07-25")),
          sell_price_am = round(week28_am[4:6]*1.4, 0), #only use thursday-saturday
          sell_price_pm = round(week28_pm[4:6]*1.25, 0), #round to make values integers
          weekday = weekdays(date), week_num = as.integer(epiweek(date)),
          week_pattern = "unknown") %>%
  print %>%
  ggplot(aes(x = date))+
  geom_line(aes(y = sell_price_am), color = 'blue', na.rm = T)+
  geom_point(aes(y = sell_price_am), color = 'blue', na.rm = T)+
  geom_line(aes(y = sell_price_pm), color = 'purple', na.rm = T)+
  geom_point(aes(y = sell_price_pm), color = 'purple', na.rm = T)+
  geom_point(aes(y = buy_price), color = 'red', na.rm = T)+
  labs(title = "Predicted prices for week 30 (scenario 1)",
       subtitle = "Red: buy price\nBlue: AM sell price\nPurple: PM sell price",
       x = "Date", y = "Price")+
  scale_x_date(date_breaks = "1 day",
               date_labels = "%b %d")
```

```
## # A tibble: 7 x 7
##   date       buy_price sell_price_am sell_price_pm weekday week_num week_pattern
##   <date>         <dbl>         <dbl>         <dbl> <chr>      <int> <chr>
## 1 2020-07-19        93            NA            NA Sunday        30 unknown
## 2 2020-07-20        NA           140           127 Monday        30 unknown
## 3 2020-07-21        NA           183           212 Tuesday       30 unknown
## 4 2020-07-22        NA           158            83 Wednes~       30 unknown
## 5 2020-07-23        NA           190           139 Thursd~       30 unknown
## 6 2020-07-24        NA           101            89 Friday        30 unknown
## 7 2020-07-25        NA            87           141 Saturd~       30 unknown
```

Let's plot week 28 again to compare:

```
turnips_s1 %>%
  filter(week_num == 28) %>%
  ggplot(aes(x = date))+
  geom_line(aes(y = sell_price_am), color = 'blue', na.rm = T)+
  geom_point(aes(y = sell_price_am), color = 'blue', na.rm = T)+
  geom_line(aes(y = sell_price_pm), color = 'purple', na.rm = T)+
  geom_point(aes(y = sell_price_pm), color = 'purple', na.rm = T)+
  geom_point(aes(y = buy_price), color = 'red', na.rm = T)+
  labs(title = "Prices for week 28",
       subtitle = "Red: buy price\nBlue: AM sell price\nPurple: PM sell price",
       x = "Date", y = "Price")+
  scale_x_date(date_breaks = "1 day",
               date_labels = "%b %d")
```



Our predictions for week 30 seem to somewhat follow the overall pattern of week 28. Additionally, for both weeks 23 and 28, a notable feature is that the Friday PM selling price is much larger than the AM selling price, which is also apparent here in week 30.

Scenario 2: Buying price is 107 bells on Sunday. Selling prices are 104/138 on Monday, 65/58 on Tuesday, 109/101 on Wednesday

Like the previous scenario, we will add new rows into our table for these values.

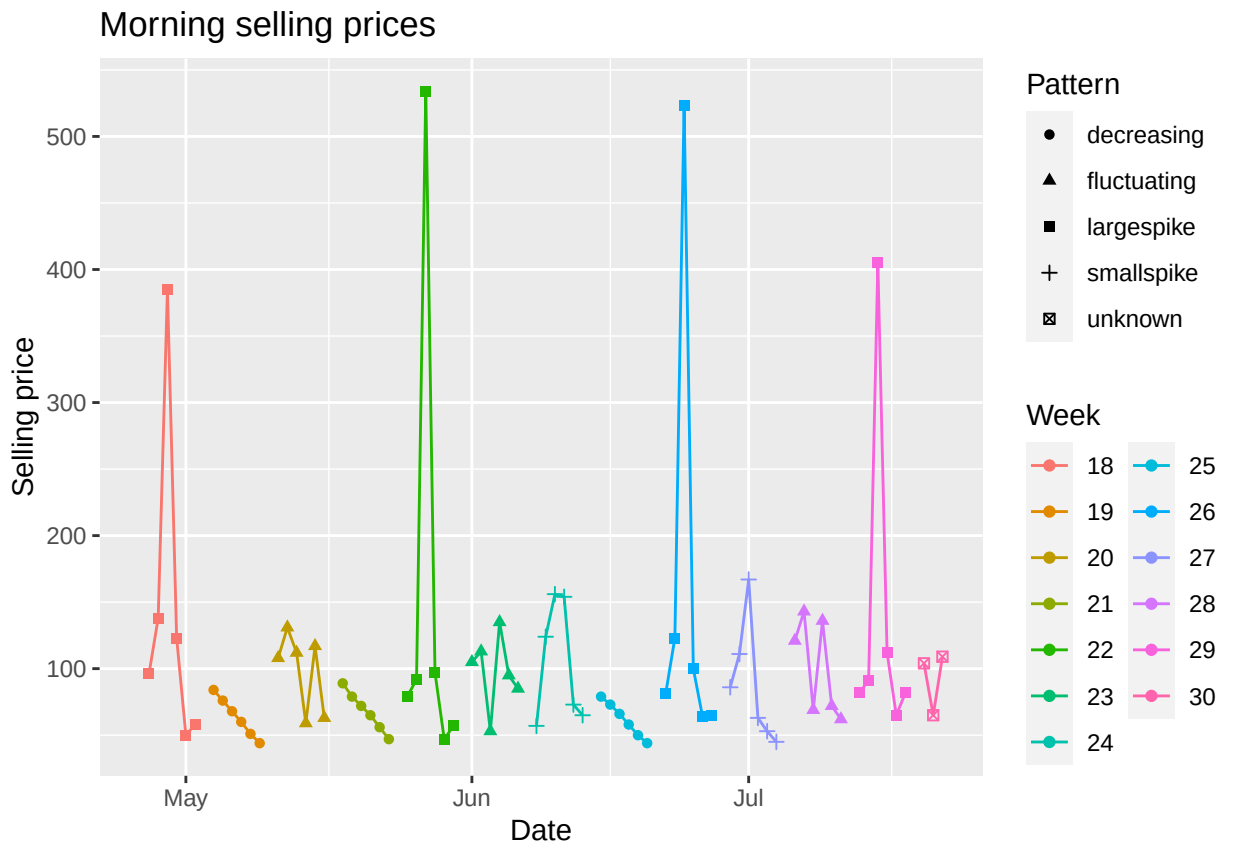
```
turnips_s2 = turnips %>%
  add_row(date = as.Date(c("2020-07-19", "2020-07-20",
                           "2020-07-21", "2020-07-22")),
          buy_price = c(107, rep(NA, 3)),
          sell_price_am = c(NA, 104, 65, 109),
          sell_price_pm = c(NA, 138, 58, 101),
          weekday = weekdays(date), week_num = as.integer(epiweek(date)),
          week_pattern = "unknown")

tail(turnips_s2, 5)
```

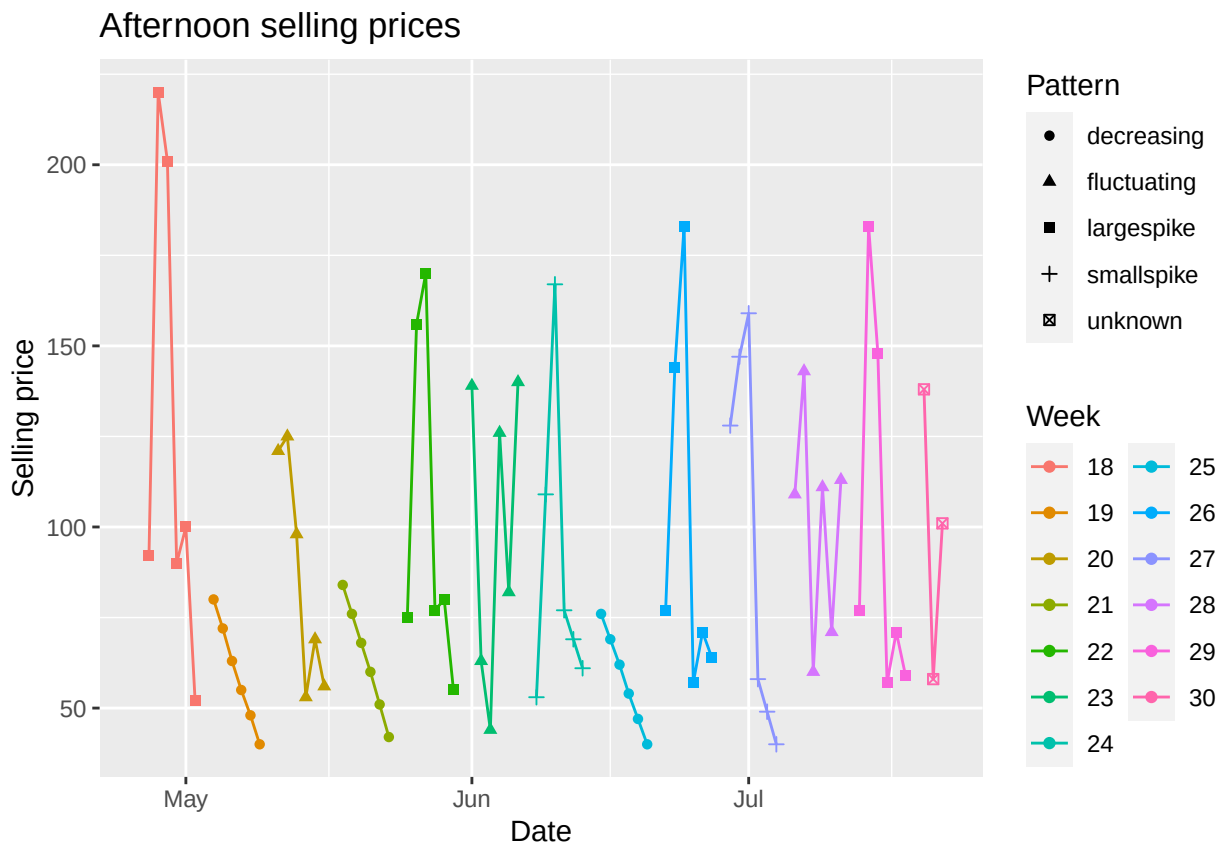
```
## # A tibble: 5 x 7
##   date       buy_price sell_price_am sell_price_pm weekday week_num week_pattern
##   <date>         <dbl>         <dbl>         <dbl> <chr>      <int> <chr>
```

```
## 1 2020-07-18         NA            82            59 Saturd~       29 largespike
## 2 2020-07-19        107            NA            NA Sunday        30 unknown
## 3 2020-07-20         NA           104           138 Monday        30 unknown
## 4 2020-07-21         NA            65            58 Tuesday       30 unknown
## 5 2020-07-22         NA           109           101 Wednes~       30 unknown
```

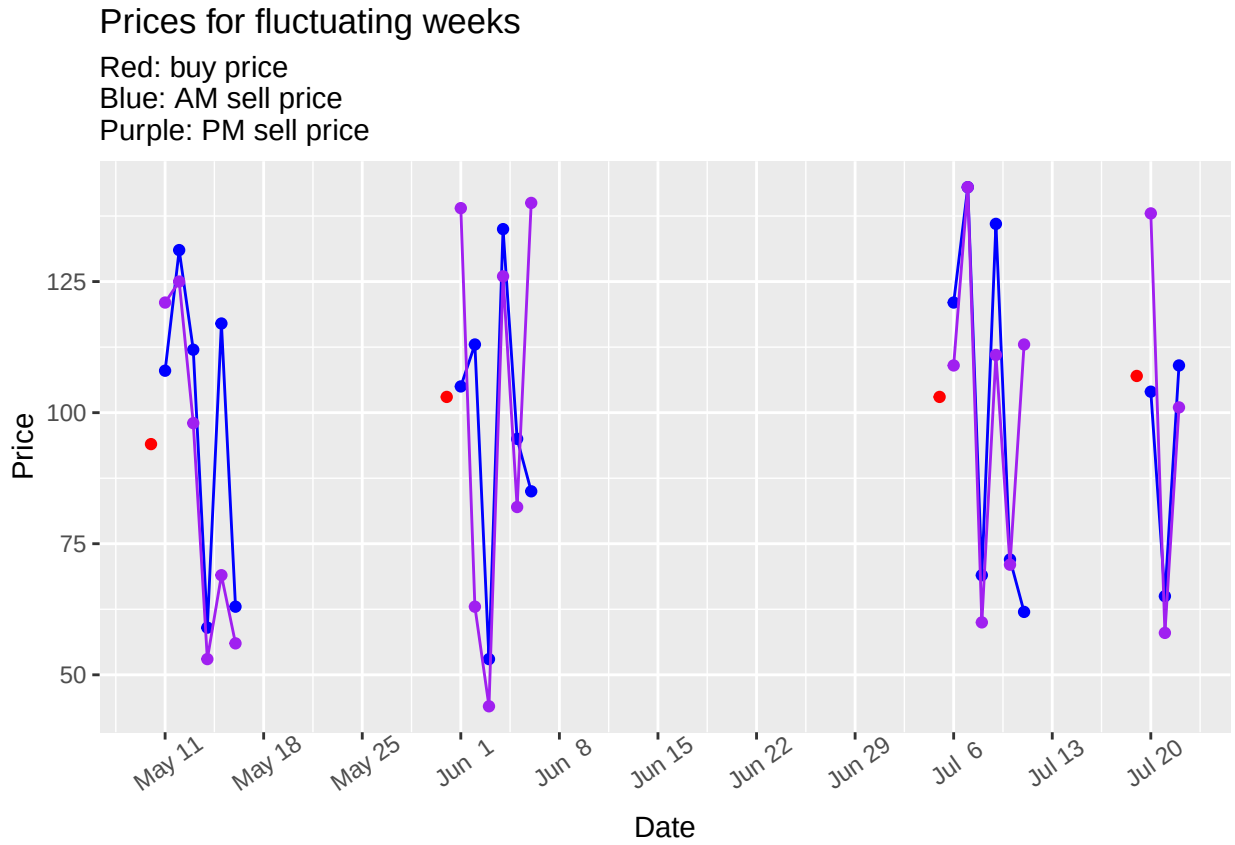
```
ggplot(turnips_s2, aes(date, sell_price_am, shape = week_pattern,
                       color = as.factor(week_num)))+
  geom_point(na.rm = T)+
  geom_line(na.rm = T)+
  labs(x = "Date", y = "Selling price", color = "Week", shape = "Pattern",
       title = "Morning selling prices")+
  guides(col = guide_legend(nrow = 7))
```




```
ggplot(turnips_s2, aes(date, sell_price_pm, shape = week_pattern,
                       color = as.factor(week_num)))+
  geom_point(na.rm = T)+
  geom_line(na.rm = T)+
  labs(x = "Date", y = "Selling price", color = "Week", shape = "Pattern",
       title = "Afternoon selling prices")+
  guides(col = guide_legend(nrow = 7))
```




```
turnips_s2 %>%
        filter(week_pattern == "fluctuating" | week_num == 30) %>%
  ggplot(aes(x = date))+
  geom_point(aes(y = buy_price), color = 'red', na.rm = T)+
  geom_line(aes(y = sell_price_am, group = week_num), color = 'blue', na.rm = T)+
  geom_point(aes(y = sell_price_am, group = week_num), color = 'blue', na.rm = T)+
  geom_line(aes(y = sell_price_pm, group = week_num), color = 'purple', na.rm = T)+
  geom_point(aes(y = sell_price_pm, group = week_num), color = 'purple', na.rm = T)+
  labs(title = "Prices for fluctuating weeks",
       subtitle = "Red: buy price\nBlue: AM sell price\nPurple: PM sell price",
       x = "Date", y = "Price")+
  scale_x_date(breaks = "1 week", date_labels = "%b %e")+
  theme(axis.text.x = element_text(angle = 35, vjust = 0.8))
```



This time, the AM selling prices do not resemble the other fluctuating weeks very much at all, but overall, but the PM selling prices somewhat resemble week 23. Overall, while week 30 seems to be its own new pattern, it bears the most resemblance to the fluctuating weeks.

We can see how the prices change day-to-day throughout the week for these three weeks, along with the small chunk of our new week. `change_am` will say "increase" if the current day's morning selling price is higher than yesterday's, and "decrease" if the current day's morning selling price is lower than yesterday's. The same idea applies to `change_pm`, except for afternoon selling prices.

```
turnips_s2 %>%
  filter(week_pattern == "fluctuating" & weekday != "Sunday" | week_num == 30) %>%
  select(weekday, week_num, sell_price_am, sell_price_pm) %>%
  group_by(week_num) %>%
  mutate(change_am = if_else(lag(sell_price_am) < sell_price_am, "increase", "decrease"),
         change_pm = if_else(lag(sell_price_pm) < sell_price_pm, "increase", "decrease")) %>%
  print(n = nrow(turnips_s2))
```

```
## # A tibble: 22 x 6
## # Groups:   week_num [4]
##    weekday   week_num sell_price_am sell_price_pm change_am change_pm
##    <chr>        <int>         <dbl>         <dbl> <chr>     <chr>
##  1 Monday          20           108           121 <NA>      <NA>
##  2 Tuesday         20           131           125 increase  increase
##  3 Wednesday       20           112            98 decrease  decrease
##  4 Thursday        20            59            53 decrease  decrease
##  5 Friday          20           117            69 increase  increase
##  6 Saturday        20            63            56 decrease  decrease
```

```
##  7 Monday          23          105          139 <NA>      <NA>
##  8 Tuesday         23          113           63 increase  decrease
##  9 Wednesday       23           53           44 decrease  decrease
## 10 Thursday        23          135          126 increase  increase
## 11 Friday          23           95           82 decrease  decrease
## 12 Saturday        23           85          140 decrease  increase
## 13 Monday          28          121          109 <NA>      <NA>
## 14 Tuesday         28          143          143 increase  increase
## 15 Wednesday       28           69           60 decrease  decrease
## 16 Thursday        28          136          111 increase  increase
## 17 Friday          28           72           71 decrease  decrease
## 18 Saturday        28           62          113 decrease  increase
## 19 Sunday          30           NA           NA <NA>      <NA>
## 20 Monday          30          104          138 <NA>      <NA>
## 21 Tuesday         30           65           58 decrease  decrease
## 22 Wednesday       30          109          101 increase  increase
```

For all of the weeks, the price increases from Monday to Tuesday in both the morning and afternoon, with the exception of week 23 having a decrease from Monday PM to Tuesday PM. However, our fictional week 30 experiences a decrease from Monday to Tuesday for both morning and afternoon. Additionally, all three fluctuating weeks decrease from Tuesday to Wednesday, but week 30 increases from Tuesday to Wednesday.

For weeks 23 and 28, Wednesday to Thursday is a price increase, while week 20 experienced a price decrease between these days. For Friday, week 20 experienced an increase while weeks 23 and 28 experienced a decrease. Every Saturday morning had a decrease from Friday morning, with Saturday afternoon a little more variable: week 20 had a decrease while weeks 23 and 28 had an increase.

While week 30's price values seem to mimic week 23 the most (starting high on Monday and decreasing), the overall pattern of price changes seems to match week 28, just inverted. That is, whenever week 28 increases, week 30 decreases, and vice versa. This is evident in both the table above and the previous plot. As such, we can expect week 30 of Thursday to have a price decrease for both the morning and afternoon.

```
turnips_s2 %>%
  filter(week_pattern == "fluctuating" & weekday %in% c("Wednesday", "Thursday")) %>%
  select(weekday, week_num, sell_price_am, sell_price_pm) %>%
  group_by(week_num) %>%
  mutate(wed_thurs_ratio_am = sell_price_am/lag(sell_price_am),
         wed_thurs_ratio_pm = sell_price_pm/lag(sell_price_pm))
```

```
## # A tibble: 6 x 6
## # Groups:   week_num [3]
##   weekday week_num sell_price_am sell_price_pm wed_thurs_ratio~ wed_thurs_ratio~
##   <chr>      <int>         <dbl>         <dbl>            <dbl>            <dbl>
## 1 Wednes~       20           112            98           NA               NA
## 2 Thursd~       20            59            53            0.527            0.541
## 3 Wednes~       23            53            44           NA               NA
## 4 Thursd~       23           135           126            2.55             2.86
## 5 Wednes~       28            69            60           NA               NA
## 6 Thursd~       28           136           111            1.97             1.85
```

Whenever prices decrease from Wednesday to Thursday, the price is about 52-54% of Wednesday's price, while price increases range anywhere from 85-186%. However, these higher increases, such as in week 23, are likely due to lower prices on Wednesday. Week 28's Wednesday AM selling price is 69 and jumps to 136 on Thursday, while week 23's Wednesday AM selling price is only 53 and jumps to 135 on Thursday morning.

Despite the lower selling price on Wednesday compared to week 28, the Thursday morning prices are nearly identical. Additionally, the afternoon selling price is consistently higher than the morning selling price.

```
turnips_s2 %>%
  filter(week_num == 30 & weekday != "Sunday") %>%
  select(weekday, sell_price_am, sell_price_pm)
```

```
## # A tibble: 3 x 3
##   weekday   sell_price_am sell_price_pm
##   <chr>             <dbl>         <dbl>
## 1 Monday              104           138
## 2 Tuesday              65            58
## 3 Wednesday           109           101
```

Because we're expecting a decrease for Thursday, we can impute similar values into Thursday's row.

```
turnips_s2 = turnips_s2 %>%
  add_row(date = as.Date("2020-07-23"),
          sell_price_am = 60, sell_price_pm = 55,
          weekday = weekdays(date), week_num = as.integer(epiweek(date)),
          week_pattern = "unknown")
```

Now that we have a way to investigate weekly patterns and price changes, let's look through the rest of the days for these weeks. `am_ratio` and `pm_ratio` are functionally the same as the previous ratios calculated, just with a different name.

```
turnips_s2 %>%
  filter(week_pattern == "fluctuating" & weekday %in% c("Thursday", "Friday", "Saturday")) %>%
  select(weekday, week_num, sell_price_am, sell_price_pm) %>%
  group_by(week_num) %>%
  mutate(am_ratio = sell_price_am/lag(sell_price_am),
         pm_ratio = sell_price_pm/lag(sell_price_pm))
```

```
## # A tibble: 9 x 6
## # Groups:   week_num [3]
##   weekday  week_num sell_price_am sell_price_pm am_ratio pm_ratio
##   <chr>       <int>         <dbl>         <dbl>    <dbl>    <dbl>
## 1 Thursday       20            59            53   NA       NA
## 2 Friday         20           117            69    1.98     1.30
## 3 Saturday       20            63            56    0.538    0.812
## 4 Thursday       23           135           126   NA       NA
## 5 Friday         23            95            82    0.704    0.651
## 6 Saturday       23            85           140    0.895    1.71
## 7 Thursday       28           136           111   NA       NA
## 8 Friday         28            72            71    0.529    0.640
## 9 Saturday       28            62           113    0.861    1.59
```

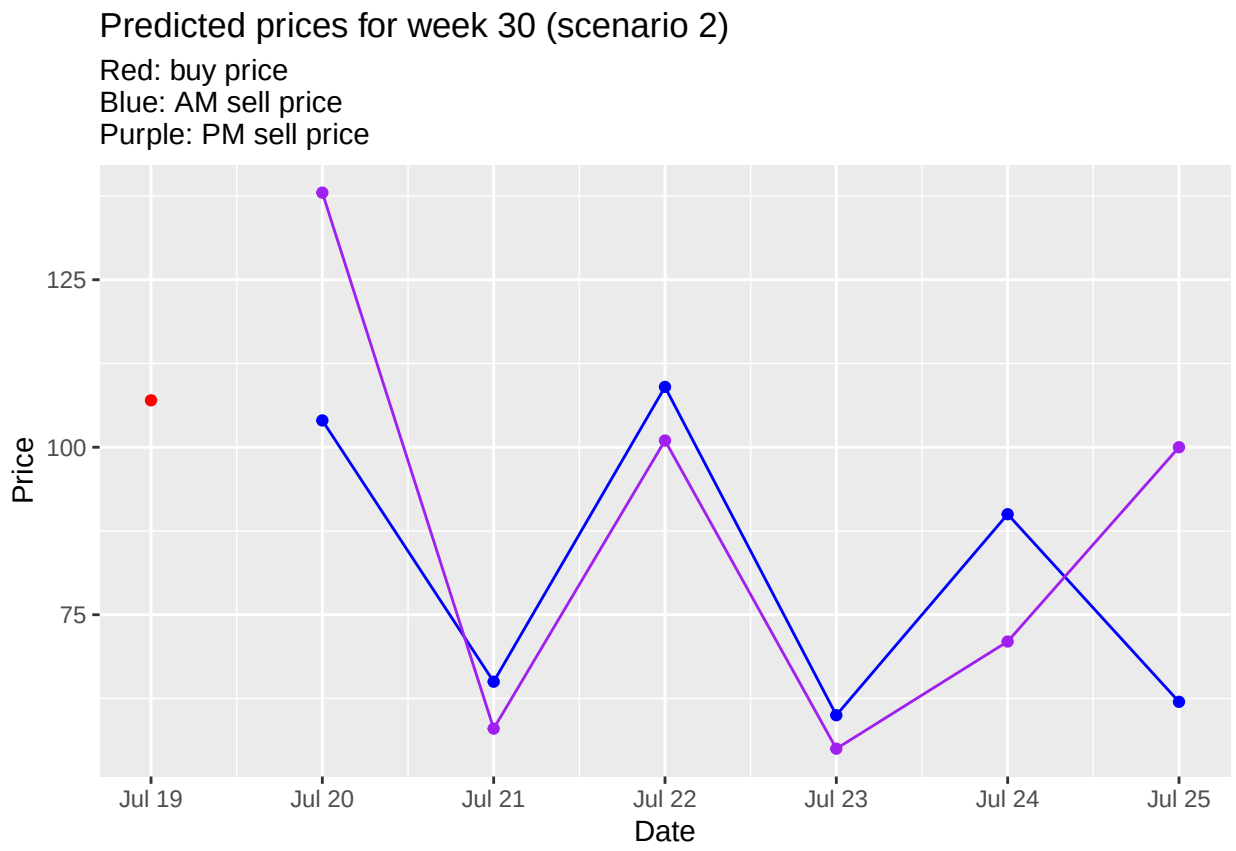
Since week 30 is presumed to be an inverse pattern of week 28, we can assume Friday's price will be higher than Thursday's. Such an occurrence only appears once, in week 20. Friday's morning price is about twice as high as Thursday's, and Friday's afternoon price is about 1.3 times as large as Thursday's afternoon price.

For Saturday, the morning selling price is always lower, ranging from 54-89% as large as Friday morning's price. The afternoon selling prices range from being about 20% smaller, like in week 20, or 1.5-1.7 times

as large, as in weeks 23 and 28. However, since week 30's overall pattern is most similar to the opposite of week 28's price changes, simply swapping week 28 is the most obvious method that comes to mind. Note that Friday and Saturday afternoons are always opposite: if Friday increases, then Saturday decreases, and vice versa.

We'll impute the new prices based on these historical changes and plot the prices:

```
turnips_s2 %>%
  add_row(date = as.Date(c("2020-07-24", "2020-07-25")),
          sell_price_am = as.integer(c(60*1.5, 60*1.5*0.7)),
          sell_price_pm = as.integer(c(55*1.3, 55*1.3*1.4)),
          weekday = weekdays(date), week_num = as.integer(epiweek(date)),
          week_pattern = "unknown") %>%
  filter(week_num == 30) %>%
  ggplot(aes(x = date))+
  geom_line(aes(y = sell_price_am), color = "blue", na.rm = T)+
  geom_point(aes(y = sell_price_am), color = "blue", na.rm = T)+
  geom_line(aes(y = sell_price_pm), color = "purple", na.rm = T)+
  geom_point(aes(y = sell_price_pm), color = "purple", na.rm = T)+
  geom_point(aes(y = buy_price), color = "red", na.rm = T)+
  labs(title = "Predicted prices for week 30 (scenario 2)",
       subtitle = "Red: buy price\nBlue: AM sell price\nPurple: PM sell price",
       x = "Date", y = "Price")+
  scale_x_date(date_breaks = "1 day",
               date_labels = "%b %d")
```



Due to how different this week was to the other fluctuating weeks, we had to rely on historical ratios

between consecutive days' prices to figure out how to price the turnips. For the most part, it seemed to be the exact opposite of week 28, but certain patterns made it difficult to replicate it perfectly. The most notable difference is how, despite this inversion, it matched week 28's afternoon price jump between Friday and Saturday. This is in agreement with week 23, as the afternoon selling price on Friday was higher than Saturday in this week as well.

While this is certainly a very ad hoc approach to predicting the week's prices, there is no specific external factor that can be used to predict what "influences" the selling prices. While other features may be generated and be useful predictors for the price, they are possibly coincidental and not truly indicative of what the price will be. For instance, using the weekday number (1 for Monday, 2 for Tuesday, so on) could be useful since it is the same across each week, but at the same time, since the prices can be quite erratic, a fancy machine learning model may or may not give us significantly better predictions.

References

[1] Kim, T. K., & Park, J. H. (2019). More about the basic assumptions of t-test: normality and sample size. *Korean journal of anesthesiology*, 72(4), 331–335. https://doi.org/10.4097/kja.d.18.00292

[2] Bewick, V., Cheek, L., & Ball, J. (2004). Statistics review 10: further nonparametric methods. *Critical care* (London, England), 8(3), 196–199. https://doi.org/10.1186/cc2857