Assignment

October 3, 2025

Contents

List of Figures

List of Tables

1 Preparations

Before messing around with the stock data, the environment should install and load the dplyr and lubridate packages to perform easier data analysis.

[3]:
```
# installation
install.packages("dplyr")
install.packages("lubridate")
install.packages("ggplot2")
```

```
Installing package into '/home/codespace/R/x86_64-pc-linux-gnu-library/4.3'
(as 'lib' is unspecified)
```

```
Installing package into '/home/codespace/R/x86_64-pc-linux-gnu-library/4.3'
(as 'lib' is unspecified)
```

```
Installing package into '/home/codespace/R/x86_64-pc-linux-gnu-library/4.3'
(as 'lib' is unspecified)
```

After installing them, the packages are loaded in.

[4]:
```
#| warning: false
library(dplyr)
library(lubridate)
library(ggplot2)
```

```
Attaching package: 'dplyr'
```

```
The following objects are masked from 'package:stats':

    filter, lag
```

```
The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
Attaching package: 'lubridate'
```

```
The following objects are masked from 'package:base':

    date, intersect, setdiff, union
```

2 Data Analysis

Now let's start actually processing and transforming the stock data. First of all, the data must be *read* into existence, so that it becomes a workable dataframe. Also to ensure correct formatting, the `datadate` variable is adjusted to the standard date format.

[5]:
```r
# load csv file as stock_data but clean the date format
stock_data = read.csv("compustat_food_bev.csv")
stock_data$datadate = as.Date(stock_data$datadate, format = "%d/%m/%Y")
```

With the stock data in place, only the relevant data set gets filtered out. Our team will focus on the following tickers: + SBUX (Starbucks) + WEN (Wendy's) + PBPB (Potbelly) + CMG (Chipotle) + DPZ (Domino's Pizza)

2.1 SBUX (Starbucks)

[ ]:
```r
#   Pick ticker & prepare base frame ------------------------------------
stock_data_sbux <- stock_data %>%
  filter(tic == "SBUX") %>%
  arrange(datadate)

#   Features 1-6 --------------------------------------------------------
stock_data_sbux <- stock_data_sbux %>%
  arrange(datadate) %>%
  mutate(
    # 1) daily return
    return_daily   = (prccd - lag(prccd)) / lag(prccd),

    # 2) 10-day momentum
    momentum_10day = prccd - lag(prccd, 10),

    # 3) daily range
    range_daily    = prchd - prcld,

    # 4) MFV (guard against divide-by-zero)
    MFV = if_else(range_daily == 0 | is.na(range_daily),
                  NA_real_,
                  (((prccd - prcld) - (prchd - prccd)) / range_daily) * cshtrd),

    # 5-6) month & year (use new names to avoid clash with year())
    month_num = lubridate::month(datadate),
    year_num  = lubridate::year(datadate)
  )

# 7) Total trading volume in June 2023
trad_vol_jun23 <- stock_data_sbux %>%
```

```r
  filter(year_num == 2023, month_num == 6)
total_volume <- sum(trad_vol_jun23$cshtrd, na.rm = TRUE)
print(paste("The total trading volume in June 2023 was",
            total_volume, "shares"))

# 8) Mean daily return over the entire period
mean_daily_return <- mean(stock_data_sbux$return_daily, na.rm = TRUE)
print(paste("The mean daily return over the entire period was",
            round(mean_daily_return*100, 2), "%"))

# 9. Day with largest positive high price
largest_high_price <- filter(stock_data_sbux, prchd == max(prchd, na.rm = TRUE))
print(paste("The day with the largest high price was on",
            largest_high_price$datadate,
            "with a price of", largest_high_price$prchd))

# https://investor.starbucks.com/news/financial-releases/news-details/2021/
 ↪Starbucks-Reports-Record-Q3-Fiscal-2021-Results/default.aspx
# Corporate context: Starbucks Q3 FY21 earnings (announced July 27, 2021)␣
 ↪reported record revenue.
print("This spike was likely driven by investor optimism ahead of record Q3␣
 ↪earnings, released on July 27, 2021.")

# 10. Day with the largest positive daily return
largest_daily_return <- filter(stock_data_sbux, return_daily ==␣
 ↪max(return_daily, na.rm = TRUE))
print(paste("The day with the largest daily return occurred on",
            largest_daily_return$datadate,
            "at", round(largest_daily_return$return_daily*100, 2), "%"))

# https://investor.starbucks.com/news/financial-releases/news-details/2022/
 ↪Starbucks-Reports-Q2-Fiscal-2022-Results/default.aspx
# Corporate context: Starbucks Q2 FY22 earnings (announced May 3, 2022) showed␣
 ↪strong revenue growth.
print("This jump was driven by Starbucks' Q2 FY22 earnings release on May 3,␣
 ↪2022, which beat expectations.")

highest_return_SBUX = stock_data_sbux$return_daily[which.
 ↪max(stock_data_sbux$return_daily)]*100
```

```
[1] "The total trading volume in June 2023 was 151045270 shares"
```

```
[1] "The mean daily return over the entire period was 0.03 %"
[1] "The day with the largest high price was on 2021-07-23 with a price of
126.32"
[1] "This spike was likely driven by investor optimism ahead of record Q3
earnings, released on July 27, 2021."
[1] "The day with the largest daily return occurred on 2022-05-04 at 9.83 %"
```

```
[1] "This jump was driven by Starbucks' Q2 FY22 earnings release on May 3, 2022,
which beat expectations."
```

2.2 WEN (Wendy's)

[ ]:
```r
# construct new data table with stock data exclusively tied to Wendy's
stock_data_wen = filter(stock_data, tic=="WEN")
```

Inbefore tackling the tasks, some new columns must be created. These are going to be loaded with metrics such as the daily returns, overnight returns, volume change and MFV of our stock data.

[ ]:
```r
# create new empty columns
stock_data_wen$retd = 0
stock_data_wen$retov = 0
stock_data_wen$volch = 0
stock_data_wen$mfv = 0
```

Next, we can actually perform the calculations for these metrics.

2.3 (1) Daily Returns

[ ]:
```
# calculate daily returns as: (closing price_t/closing price_t-1)-1
head(stock_data_wen <- mutate(stock_data_wen, retd = round((prccd/
  ↪lag(prccd))-1, 2)))
```

A data.frame: 6 × 13

	datadate <date>	tic <chr>	cshtrd <int>	prccd <dbl>	prchd <dbl>	prcld <dbl>	prcod <dbl>	exchg <int>	sic <int>
1	2020-09-01	WEN	2929911	20.950	21.16	20.7400	20.94	14	5812
2	2020-09-02	WEN	3814903	21.970	22.08	21.0100	21.01	14	5812
3	2020-09-03	WEN	4280982	21.950	22.48	21.6450	21.95	14	5812
4	2020-09-04	WEN	3351921	21.580	22.38	21.4050	22.20	14	5812
5	2020-09-08	WEN	3439170	21.850	22.05	21.2300	21.43	14	5812
6	2020-09-09	WEN	2847727	22.495	22.61	21.8322	22.00	14	5812

2.4 (2) Overnight Returns

[ ]:
```
# calculate overnight returns as: (openning price_t/closing price_t-1)-1
head(stock_data_wen <- mutate(stock_data_wen, retov = round((prcod/
  ↪lag(prccd))-1, 2)))
```

A data.frame: 6 × 13

	datadate <date>	tic <chr>	cshtrd <int>	prccd <dbl>	prchd <dbl>	prcld <dbl>	prcod <dbl>	exchg <int>	sic <int>
1	2020-09-01	WEN	2929911	20.950	21.16	20.7400	20.94	14	5812
2	2020-09-02	WEN	3814903	21.970	22.08	21.0100	21.01	14	5812
3	2020-09-03	WEN	4280982	21.950	22.48	21.6450	21.95	14	5812
4	2020-09-04	WEN	3351921	21.580	22.38	21.4050	22.20	14	5812
5	2020-09-08	WEN	3439170	21.850	22.05	21.2300	21.43	14	5812
6	2020-09-09	WEN	2847727	22.495	22.61	21.8322	22.00	14	5812

2.5 (3) Volume Change

[ ]:
```
# calculate volume change as: (volume_t/volume_t-1)-1
head(stock_data_wen <- mutate(stock_data_wen, volch = round((cshtrd/
  ↪lag(cshtrd))-1, 2)))
```

A data.frame: 6 × 13

	datadate <date>	tic <chr>	cshtrd <int>	prccd <dbl>	prchd <dbl>	prcld <dbl>	prcod <dbl>	exchg <int>	sic <int>
1	2020-09-01	WEN	2929911	20.950	21.16	20.7400	20.94	14	5812
2	2020-09-02	WEN	3814903	21.970	22.08	21.0100	21.01	14	5812
3	2020-09-03	WEN	4280982	21.950	22.48	21.6450	21.95	14	5812
4	2020-09-04	WEN	3351921	21.580	22.38	21.4050	22.20	14	5812
5	2020-09-08	WEN	3439170	21.850	22.05	21.2300	21.43	14	5812
6	2020-09-09	WEN	2847727	22.495	22.61	21.8322	22.00	14	5812

2.6 (4) Money Flow Volume Indicator (MFV)

[ ]:
```
# calculate mfv
head(stock_data_wen <- mutate(stock_data_wen,
    mfv = round((
        ((prccd-prcld)-(prchd-prcld))/(prchd-prcld)
        ), 2)
))
```

A data.frame: 6 × 13

	datadate <date>	tic <chr>	cshtrd <int>	prccd <dbl>	prchd <dbl>	prcld <dbl>	prcod <dbl>	exchg <int>	sic <int>
1	2020-09-01	WEN	2929911	20.950	21.16	20.7400	20.94	14	5812
2	2020-09-02	WEN	3814903	21.970	22.08	21.0100	21.01	14	5812
3	2020-09-03	WEN	4280982	21.950	22.48	21.6450	21.95	14	5812
4	2020-09-04	WEN	3351921	21.580	22.38	21.4050	22.20	14	5812
5	2020-09-08	WEN	3439170	21.850	22.05	21.2300	21.43	14	5812
6	2020-09-09	WEN	2847727	22.495	22.61	21.8322	22.00	14	5812

3 More features, metrics and dates

After completing the initial analysis, the next part will cover various kinds of data manipulation and appending new variables to the dataframe.

3.1 (5) Month & (6) Year

Once more, first a few more columns are added to hold the data, after which the `month()` and `year()` functions help identifying the respective data points.

[ ]:
```
# Add column that indicate the month and year
stock_data_wen$mon = 0
stock_data_wen$yr = 0
head(stock_data_wen <- mutate(stock_data_wen,
    mon = month(datadate),
    yr = year(datadate)
))
```

A data.frame: 6 × 15

	datadate <date>	tic <chr>	cshtrd <int>	prccd <dbl>	prchd <dbl>	prcld <dbl>	prcod <dbl>	exchg <int>	sic <int>
1	2020-09-01	WEN	2929911	20.950	21.16	20.7400	20.94	14	5812
2	2020-09-02	WEN	3814903	21.970	22.08	21.0100	21.01	14	5812
3	2020-09-03	WEN	4280982	21.950	22.48	21.6450	21.95	14	5812
4	2020-09-04	WEN	3351921	21.580	22.38	21.4050	22.20	14	5812
5	2020-09-08	WEN	3439170	21.850	22.05	21.2300	21.43	14	5812
6	2020-09-09	WEN	2847727	22.495	22.61	21.8322	22.00	14	5812

3.2 (7) Total Trading Volume in June 2023

By employing a method innate to R, the `sum()` function, together with a condition tied to the `mon` and `yr` variables lends a quick result to the task.

[ ]:
```
# Calculate the total trading volume, in June 2023.
tv0623 = sum(stock_data_wen$cshtrd[
    stock_data_wen$mon == 6 & stock_data_wen$yr==2023
    ])
tv0623
```

54557454

3.3 (8) Mean Daily Return

Another method of R, the `mean()` function, produces an even quicker result to this task.

[ ]:
```
# Calculate the mean daily return, over the entire period.
return_daily_mean = mean(stock_data_wen$retd, na.rm = TRUE)*100
print(paste0(round(return_daily_mean, 2), "%"))
```

```
[1] "0.02%"
```

3.4 (9) Date with the largest High Price

The desired date relates to the maximum value across the `prchd` variable, thereby hinting at applying the `which.max()` function for indexing. Note that applying just the `max()` function does not give the desired result, since it returns the actual price instead of the row index.

[ ]:
```
# Calculate the date that saw the largest positive high price.
date_highest_high = stock_data_wen$datadate[
    which.max(stock_data_wen$prchd)
    ]
date_highest_high
```

2021-06-08

But it's still interesting to see the price.

[ ]:
```
# Said price is how much?
price_highest = stock_data_wen$prchd[which.max(stock_data_wen$prchd)]
price_highest
```

29.46

3.5 (10) Date with the largest Daily Return

This desired date relates to the maximum value across the `retd` variable, again hinting at the `which.max()` function.

[ ]:
```
# Calculate the date that saw the largest positive daily return.
date_highest_daily_return = stock_data_wen$datadate[
    which.max(stock_data_wen$retd)
    ]
date_highest_daily_return
```

2021-06-08

The return is..

[ ]:
```
# Said return is how much?
retd_highest_p = stock_data_wen$retd[which.max(stock_data_wen$retd)]*100
print(paste0(round(retd_highest_p, 2), "%"))
```

```
[1] "26%"
```

3.6 PBPB (Potbelly)

[50]:
```
#head(stock_data)
stock_data_PBPB <- filter(stock_data, tic == "PBPB")


#adding returns
stock_data_PBPB <- mutate(stock_data_PBPB, return = prccd/lag(prccd,1)-1)
```

```r
#ading overnight returns
stock_data_PBPB <- mutate(stock_data_PBPB, return_ovn = prcod/lag(prccd,1)-1)
#adding price delta intraday
stock_data_PBPB <- mutate(stock_data_PBPB, px_delta_intra = prchd-prcld)
#adding Money FLow Volume Indicator (MFV)
stock_data_PBPB <- mutate(stock_data_PBPB, MFV = ((prccd-prcld)-(prchd-prccd))/
 ↪px_delta_intra)
#head(stock_data_PBPB)

#adding the month (lubridate already part of tidyverse)

stock_data_PBPB <- mutate(stock_data_PBPB, Month = month(datadate))
stock_data_PBPB <- mutate(stock_data_PBPB, Year = year(datadate))
#head(stock_data_PBPB)

##display specific trading metrics
#Volume traded in June 2023
trad_vol_jun23 <- filter(stock_data_PBPB, Month == 6 & Year == 2023)
head(trad_vol_jun23)
print(paste(sum(trad_vol_jun23$cshtrd), " Total Common Shares Traded in June␣
 ↪2023"))

#Average return over entire period
mean_daily_return <- mean(stock_data_PBPB$return, na.rm = TRUE)
print(paste("The mean Daily Return during the entire peroid was ",␣
 ↪round(mean_daily_return*100, 2),"%"))

#Day with largest positive high price

largest_high_price <- filter(stock_data_PBPB, prchd == max(prchd))
head(largest_high_price)
print(paste("The day wiht the largest high price was on",␣
 ↪largest_high_price$datadate, "with a price of", largest_high_price$prchd))

#Day with the largest positive daily return



largest_daily_return_date_PBPB = stock_data_PBPB$datadate[
    which.max(stock_data_PBPB$return)
    ]

largest_daily_return_PBPB = stock_data_PBPB$return[which.
 ↪max(stock_data_PBPB$return)]*100
print(paste("The Day with the largest daily return occured on",␣
 ↪largest_daily_return_date_PBPB, "at", round(largest_daily_return_PBPB, 2),␣
 ↪"%"))
```

A data.frame: 6 × 18

	GVKEY <int>	iid <int>	datadate <date>	tic <chr>	conm <chr>	cshtrd <int>	prccd <dbl>	prch <db
1	18839	1	2023-06-01	PBPB	POTBELLY CORP	115522	7.98	8.05
2	18839	1	2023-06-02	PBPB	POTBELLY CORP	152263	8.32	8.41
3	18839	1	2023-06-05	PBPB	POTBELLY CORP	132449	8.10	8.25
4	18839	1	2023-06-06	PBPB	POTBELLY CORP	174599	8.37	8.55
5	18839	1	2023-06-07	PBPB	POTBELLY CORP	268521	8.72	8.82
6	18839	1	2023-06-08	PBPB	POTBELLY CORP	240227	8.21	8.91

```
[1] "6780601  Total Common Shares Traded in June 2023"
[1] "The mean Daily Return during the entire peroid was  0.13 %"
```

A data.frame: 1 × 18

	GVKEY <int>	iid <int>	datadate <date>	tic <chr>	conm <chr>	cshtrd <int>	prccd <dbl>	prch <db
1	18839	1	2023-04-26	PBPB	POTBELLY CORP	676321	10.85	11.1

```
[1] "The day wiht the largest high price was on 2023-04-26 with a price of
11.14"
[1] "The Day with the largest daily return occured on 2021-03-15 at 17.59 %"
```

3.7 CMG (Chipotle)

[45]:
```
#head(stock_data)
stock_data_CMG <- filter(stock_data, tic == "CMG")

# 1. Adding a daily return column
stock_data_CMG <- mutate(stock_data_CMG, return_daily = prccd/lag(prccd,1)-1)

# 2. Adding a volume change column
stock_data_CMG <- mutate(stock_data_CMG, change_volume = cshtrd - lag(cshtrd,␣
↪1))

# 3. Adding a 10-day momentum indicator column
stock_data_CMG <- mutate(stock_data_CMG, momentum_10day = prccd - lag(prccd,␣
↪10))

# 4. Adding a Money FLow Volume Indicator (MFV) column
stock_data_CMG <- mutate(stock_data_CMG, MFV = ((prccd-prcld)-(prchd-prccd))/
↪prchd-prcld)

#head(stock_data_CMG)

# 5. Adding a column for the month
stock_data_CMG <- mutate(stock_data_CMG, Month = month(datadate))

# 6. Adding a column for the year
stock_data_CMG <- mutate(stock_data_CMG, Year = year(datadate))
```

```r
#head(stock_data_CMG)

# 7. Total volume traded in June 2023
trad_vol_jun23 <- filter(stock_data_CMG, Month == 6 & Year == 2023)
head(trad_vol_jun23)
print(paste("The total trading volume in June 2023 was ",␣
 ↪sum(trad_vol_jun23$cshtrd)))

# 8. Mean daily return over entire period
mean_daily_return <- mean(stock_data_CMG$return, na.rm = TRUE)
print(paste("The mean daily return during the entire period was ",␣
 ↪round(mean_daily_return*100, 2),"%"))

# 9. Day with largest positive high price

largest_high_price <- filter(stock_data_CMG, prchd == max(prchd))
head(largest_high_price)
print(paste("The day with the largest high price was on",␣
 ↪largest_high_price$datadate, "with a price of", largest_high_price$prchd))

#https://newsroom.chipotle.com/
 ↪2023-07-19-BOWLS-FOR-GOALS-IS-BACK-CHIPOTLE-TO-GIVE-AWAY-FREE-ENTREES-WHEN-THE-U-S-WOMENS-N
# On July 29, 2023, Chipotle gave free entrees for every goal scored by the US␣
 ↪Women's National Team
print(paste("On July 29, 2023, Chipotle gave free entrees for every goal scored␣
 ↪by the US Women's National Team."))

# 10. Day with the largest positive daily return
largest_daily_return_CMG <- filter(stock_data_CMG, return_daily ==␣
 ↪max(return_daily, na.rm = TRUE))
head(largest_daily_return_CMG)
print(paste("The Day with the largest daily return occured on",␣
 ↪largest_daily_return_CMG$datadate, "at",␣
 ↪round(largest_daily_return_CMG$return_daily*100, 2), "%"))

#https://newsroom.chipotle.com/
 ↪2022-07-26-CHIPOTLE-ANNOUNCES-SECOND-QUARTER-2022-RESULTS
# On July 26, 2022, Chipotle announced its second quarter 2022 results
print(paste("On July 26 2022, Chipotle announced a strong Q2 performance␣
 ↪despite inflation and consumer uncertainty."))

highest_return_CMG = stock_data_CMG$return_daily[which.
 ↪max(stock_data_CMG$return_daily)]*100
```

A data.frame: 6 × 18

	GVKEY <int>	iid <int>	datadate <date>	tic <chr>	conm <chr>	cshtr <int>
1	165914	1	2023-06-01	CMG	CHIPOTLE MEXICAN GRILL INC	18754
2	165914	1	2023-06-02	CMG	CHIPOTLE MEXICAN GRILL INC	24542
3	165914	1	2023-06-05	CMG	CHIPOTLE MEXICAN GRILL INC	23306
4	165914	1	2023-06-06	CMG	CHIPOTLE MEXICAN GRILL INC	17759
5	165914	1	2023-06-07	CMG	CHIPOTLE MEXICAN GRILL INC	30554
6	165914	1	2023-06-08	CMG	CHIPOTLE MEXICAN GRILL INC	19440

```
[1] "The total trading volume in June 2023 was  5392605"
[1] "The mean daily return during the entire period was  0.07 %"
```

A data.frame: 1 × 18

	GVKEY <int>	iid <int>	datadate <date>	tic <chr>	conm <chr>	cshtr <int>
1	165914	1	2023-07-19	CMG	CHIPOTLE MEXICAN GRILL INC	23510

```
[1] "The day with the largest high price was on 2023-07-19 with a price of
2175.01"
[1] "On July 29, 2023, Chipotle gave free entrees for every goal scored by the
US Women's National Team."
```

A data.frame: 1 × 18

	GVKEY <int>	iid <int>	datadate <date>	tic <chr>	conm <chr>	cshtr <int>
1	165914	1	2022-07-27	CMG	CHIPOTLE MEXICAN GRILL INC	14310

```
[1] "The Day with the largest daily return occured on 2022-07-27 at 14.7 %"
[1] "On July 26 2022, Chipotle announced a strong Q2 performance despite
inflation and consumer uncertainty."
```

3.8 DPZ (Domino's Pizza)

[47]:
```
#head(stock_data)
stock_data_DPZ <- filter(stock_data, tic == "DPZ")

# 1. Add a column with the daily return
stock_data_DPZ <- mutate(stock_data_DPZ, return = prccd/lag(prccd,1) - 1)

# 2. Add a column with the volume change
stock_data_DPZ <- mutate(stock_data_DPZ, change_volume = cshtrd - lag(cshtrd,1))

# 3. Add a column with the close-open change
stock_data_DPZ <- mutate(stock_data_DPZ, change_close_open = prccd - prcod)

# 4. Add a column with the money flow volume indicator (MFV)
stock_data_DPZ <- mutate(stock_data_DPZ, MFV = ((prccd - prcld) - (prchd -␣
 ↪prccd)) / (prchd - prcld) * cshtrd)

# 5. Add a column the month
stock_data_DPZ <- mutate(stock_data_DPZ, Month = month(datadate))
```

```r
# 6. Add a column the year
stock_data_DPZ <- mutate(stock_data_DPZ, Year = year(datadate))
#head(stock_data_DPZ)

# 7. The total traing volume in June 2023
trad_vol_jun23 <- filter(stock_data_DPZ, Month == 6 & Year == 2023)
head(trad_vol_jun23)
print(paste(sum(trad_vol_jun23$cshtrd, na.rm = TRUE), " Total Common Shares␣
 ↪Traded in June 2023"))

# 8. Mean return over entire period
mean_daily_return <- mean(stock_data_DPZ$return, na.rm = TRUE)
print(paste("The mean Daily Return during the entire period was ",␣
 ↪round(mean_daily_return * 100, 2), "%"))

# 9. Date with largest positive high price
largest_high_price <- filter(stock_data_DPZ, prchd == max(prchd, na.rm = TRUE))
head(largest_high_price)
print(paste("The day with the largest high price was on",␣
 ↪largest_high_price$datadate, "with a price of", largest_high_price$prchd))

# 10. Date with the largest positive daily return
largest_daily_return_DPZ <- filter(stock_data_DPZ, return == max(return, na.rm␣
 ↪= TRUE))
head(largest_daily_return_DPZ)
print(paste("The day with the largest daily return occurred on",␣
 ↪largest_daily_return_DPZ$datadate, "at",␣
 ↪round(largest_daily_return_DPZ$return * 100, 2), "%"))

# https://ir.dominos.com/news-releases/news-release-details/
 ↪dominos-pizzar-announces-second-quarter-2021-financial-resultschipo
# On July 21nd 2022, Domino's Pizza Announces Second Quarter 2021 Financial␣
 ↪Results
print(paste("On July 22nd 2021, Domino's Pizza Announces a quite strong second␣
 ↪quarter of 2021 financial results"))

highest_return_DPZ = stock_data_DPZ$return[which.max(stock_data_DPZ$return)]*100
```

A data.frame: 6 × 18

	GVKEY <int>	iid <int>	datadate <date>	tic <chr>	conm <chr>	cshtrd <int>	prccd <dbl>
1	160211	2	2023-06-01	DPZ	DOMINO'S PIZZA INC	770814	296.67
2	160211	2	2023-06-02	DPZ	DOMINO'S PIZZA INC	626109	303.19
3	160211	2	2023-06-05	DPZ	DOMINO'S PIZZA INC	455485	297.83
4	160211	2	2023-06-06	DPZ	DOMINO'S PIZZA INC	510577	297.74
5	160211	2	2023-06-07	DPZ	DOMINO'S PIZZA INC	407951	297.56
6	160211	2	2023-06-08	DPZ	DOMINO'S PIZZA INC	459922	299.90

```
[1] "13772765  Total Common Shares Traded in June 2023"
[1] "The mean Daily Return during the entire period was  0.01 %"
```

A data.frame: 1 × 18

	GVKEY <int>	iid <int>	datadate <date>	tic <chr>	conm <chr>	cshtrd <int>	prccd <dbl>
1	160211	2	2021-12-31	DPZ	DOMINO'S PIZZA INC	297411	564.33

```
[1] "The day with the largest high price was on 2021-12-31 with a price of
567.57"
```

A data.frame: 1 × 18

	GVKEY <int>	iid <int>	datadate <date>	tic <chr>	conm <chr>	cshtrd <int>	prccd <dbl>
1	160211	2	2021-07-22	DPZ	DOMINO'S PIZZA INC	3034394	538.82

```
[1] "The day with the largest daily return occurred on 2021-07-22 at 14.55 %"
[1] "On July 22nd 2021, Domino's Pizza Announces a quite strong second quarter
of 2021 financial results"
```

4 Part 2 Visualisations

4.1 Task 1:

Visualise the number of tickers on each exchange that have had at least one trading day with a volume of more than 100000.

[ ]:
```r
head(stock_data)

#creating df with tickers on each exchange with one trading day of more than␣
 ↪100000
df_ticker_hvolume <- filter(stock_data, cshtrd>100000)
head(df_ticker_hvolume)

df_exch_tickr <- summarise(group_by(stock_data, exchg), unique_tic =␣
 ↪n_distinct(tic))

head(df_exch_tickr)

#plotting no of unique tickers on exchange with daily trading volume >100'000
ggplot(df_exch_tickr, aes(factor(exchg), unique_tic)) +
  geom_bar(stat = "identity", color = "darkblue", fill = "darkblue") +
  geom_text(aes(label = unique_tic),
            vjust = -0.3,              # position above the bar
            size = 5) +                # text size
  labs(title = "Number of distinct tickers per exchange\nwith daily trading␣
 ↪volume exceeding 100,000",
  x= "Exchange", y="Number of Tickers") +
  theme(plot.title = element_text(hjust = 0.5))
```

A data.frame: 6 × 12

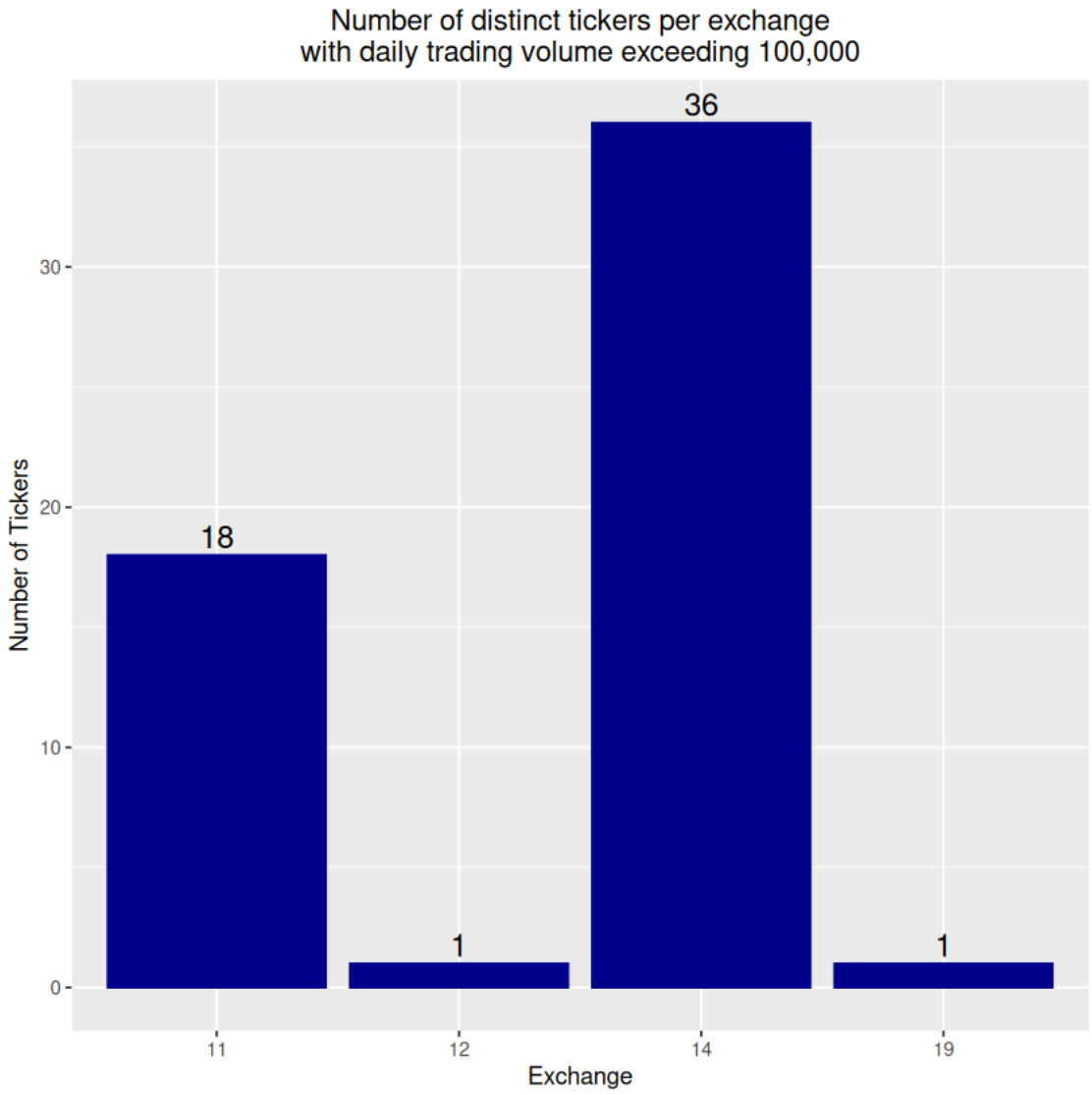
	GVKEY	iid	datadate	tic	conm	csht
	<int>	<int>	<date>	<chr>	<chr>	<in
1	186785	1	2020-09-01	ARCO	ARCOS DORADOS HOLDINGS INC	813
2	186785	1	2020-09-02	ARCO	ARCOS DORADOS HOLDINGS INC	518
3	186785	1	2020-09-03	ARCO	ARCOS DORADOS HOLDINGS INC	947
4	186785	1	2020-09-04	ARCO	ARCOS DORADOS HOLDINGS INC	534
5	186785	1	2020-09-08	ARCO	ARCOS DORADOS HOLDINGS INC	669
6	186785	1	2020-09-09	ARCO	ARCOS DORADOS HOLDINGS INC	115

A data.frame: 6 × 12

	GVKEY	iid	datadate	tic	conm	csht
	<int>	<int>	<date>	<chr>	<chr>	<in
1	186785	1	2020-09-01	ARCO	ARCOS DORADOS HOLDINGS INC	813
2	186785	1	2020-09-02	ARCO	ARCOS DORADOS HOLDINGS INC	518
3	186785	1	2020-09-03	ARCO	ARCOS DORADOS HOLDINGS INC	947
4	186785	1	2020-09-04	ARCO	ARCOS DORADOS HOLDINGS INC	534
5	186785	1	2020-09-08	ARCO	ARCOS DORADOS HOLDINGS INC	669
6	186785	1	2020-09-09	ARCO	ARCOS DORADOS HOLDINGS INC	115

A tibble: 4 × 2

exchg	unique_tic
<int>	<int>
11	18
12	1
14	36
19	1



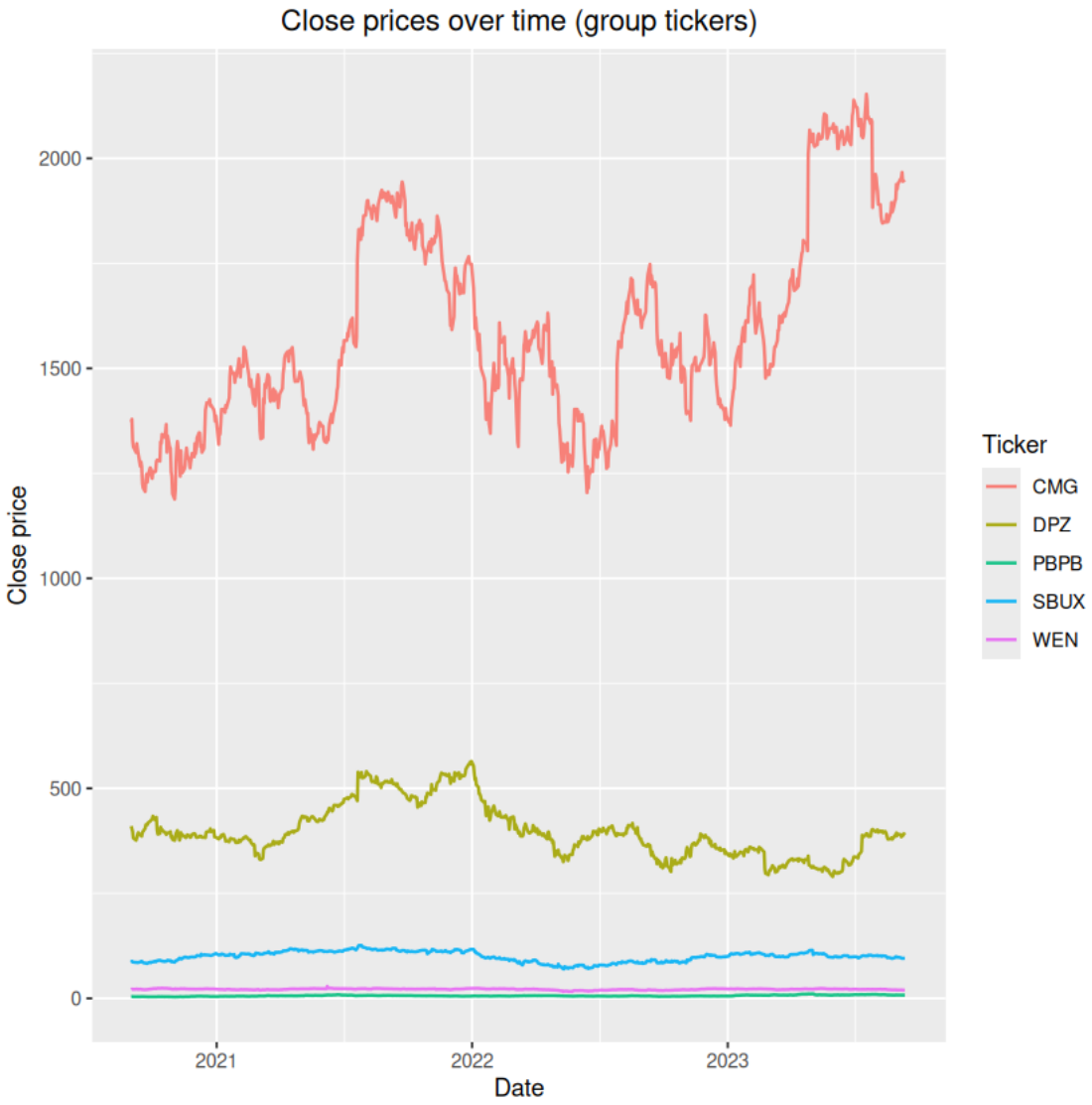
4.2 Task 2

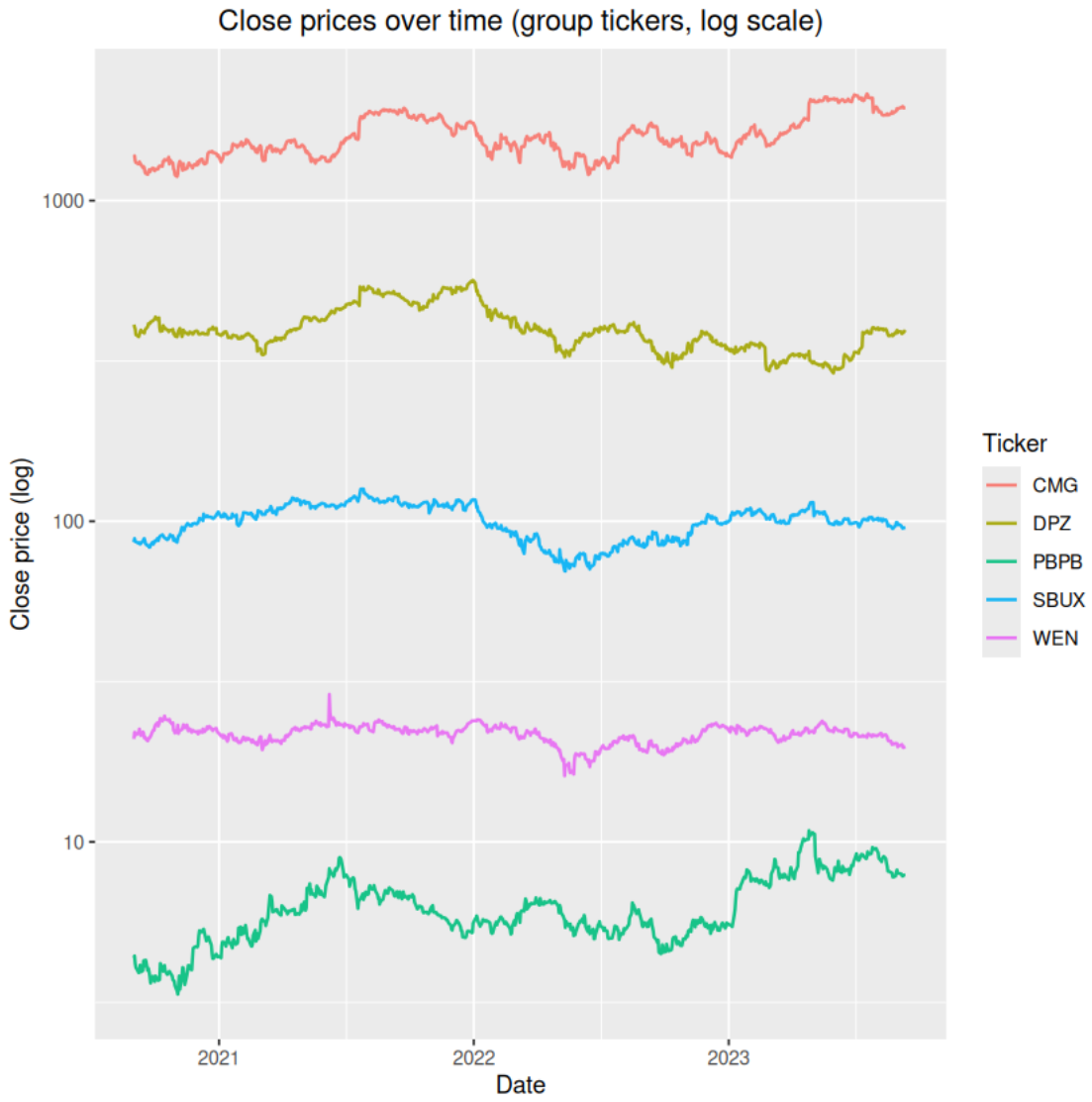
Visualise on one line plot the close prices of each ticker, over the period.

[31]:
```
# step 2

# subset of our tickers
group_tickers <- c("SBUX", "WEN", "PBPB", "CMG", "DPZ")

df_group <- stock_data %>%
  filter(tic %in% group_tickers) %>%
  arrange(datadate)
```

```r
# one line plot: close prices of each ticker over the period
ggplot(df_group, aes(x = datadate, y = prccd, color = tic)) +
  geom_line(size = 0.7, alpha = 0.9) +
  labs(
    title = "Close prices over time (group tickers)",
    x = "Date", y = "Close price", color = "Ticker"
  ) +
  theme(plot.title = element_text(hjust = 0.5))

# CMG prices are big numbers, thus use log
ggplot(df_group, aes(x = datadate, y = prccd, color = tic)) +
  geom_line(size = 0.7, alpha = 0.9) +
  scale_y_log10() +                              # <-- compresses CMG's big␣
 ↪numbers
  labs(
    title = "Close prices over time (group tickers, log scale)",
    x = "Date", y = "Close price (log)", color = "Ticker"
  ) +
  theme(plot.title = element_text(hjust = 0.5))
```

```
Warning message:
"Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
 Please use `linewidth` instead."
Warning message:
"Removed 1 row containing missing values or values outside the scale
range
(`geom_line()`)."
Warning message:
"Removed 1 row containing missing values or values outside the scale
range
(`geom_line()`)."
```


Close prices over time (group tickers)
Close price
2000
1500
1000
500
0
2021
2022
2023
Date
Ticker
CMG
DPZ
PBPB
SBUX
WEN



4.3 Task 3

consider only the ticker you analysed with the highest mean daily return over the period and...

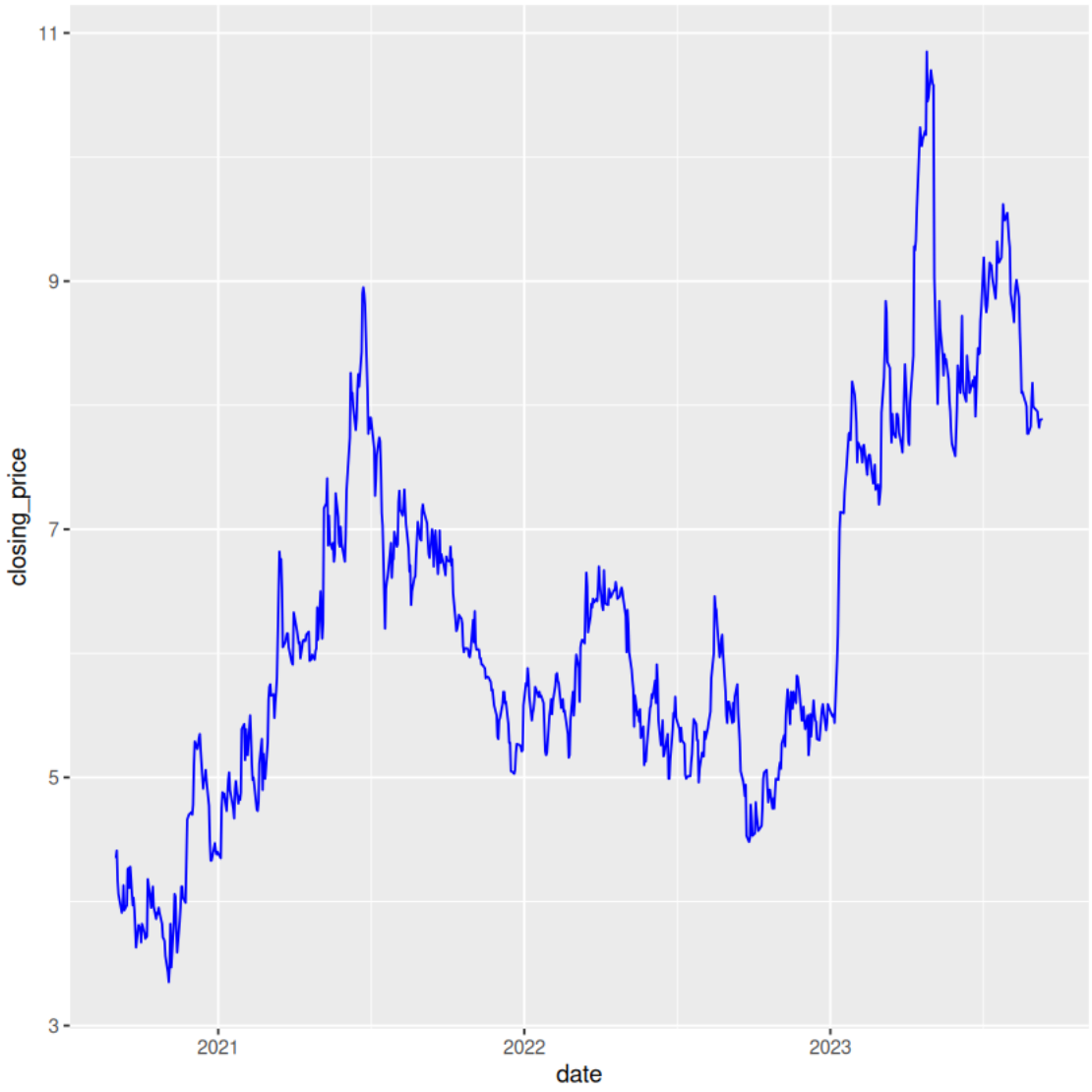
Visualise on one line plot the high and low prices, in the year 2021.

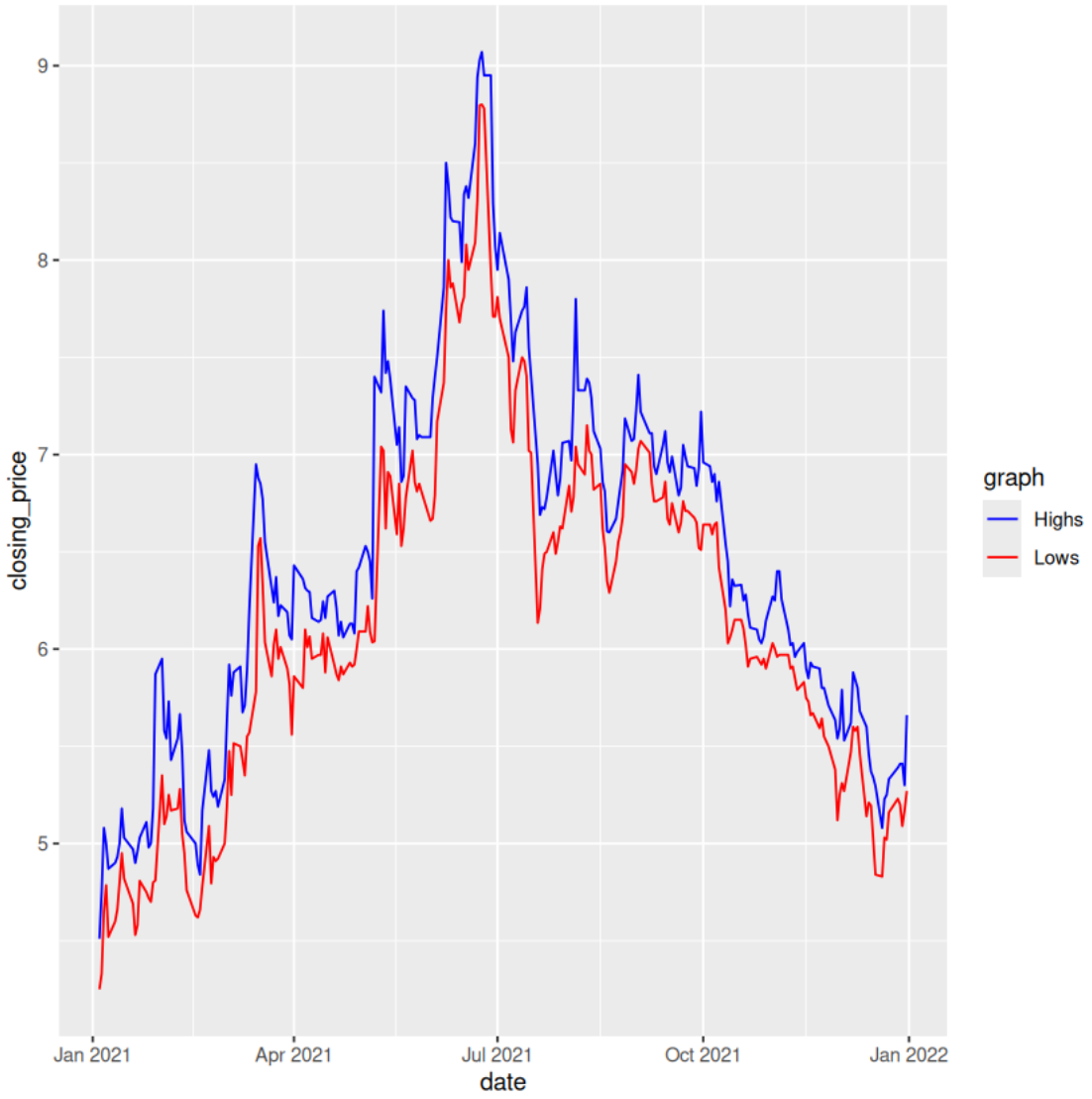
[53]:
```
# Step 3 WIP

returns_vec <- c(PBPB = largest_daily_return_PBPB, SBUX = highest_return_SBUX,␣
↪CMG = highest_return_CMG, DPZ = highest_return_DPZ)
head(returns_vec)
ticker_largest_d_return <- names(which.max(returns_vec))
value_largest_d_return <- max(returns_vec)
```

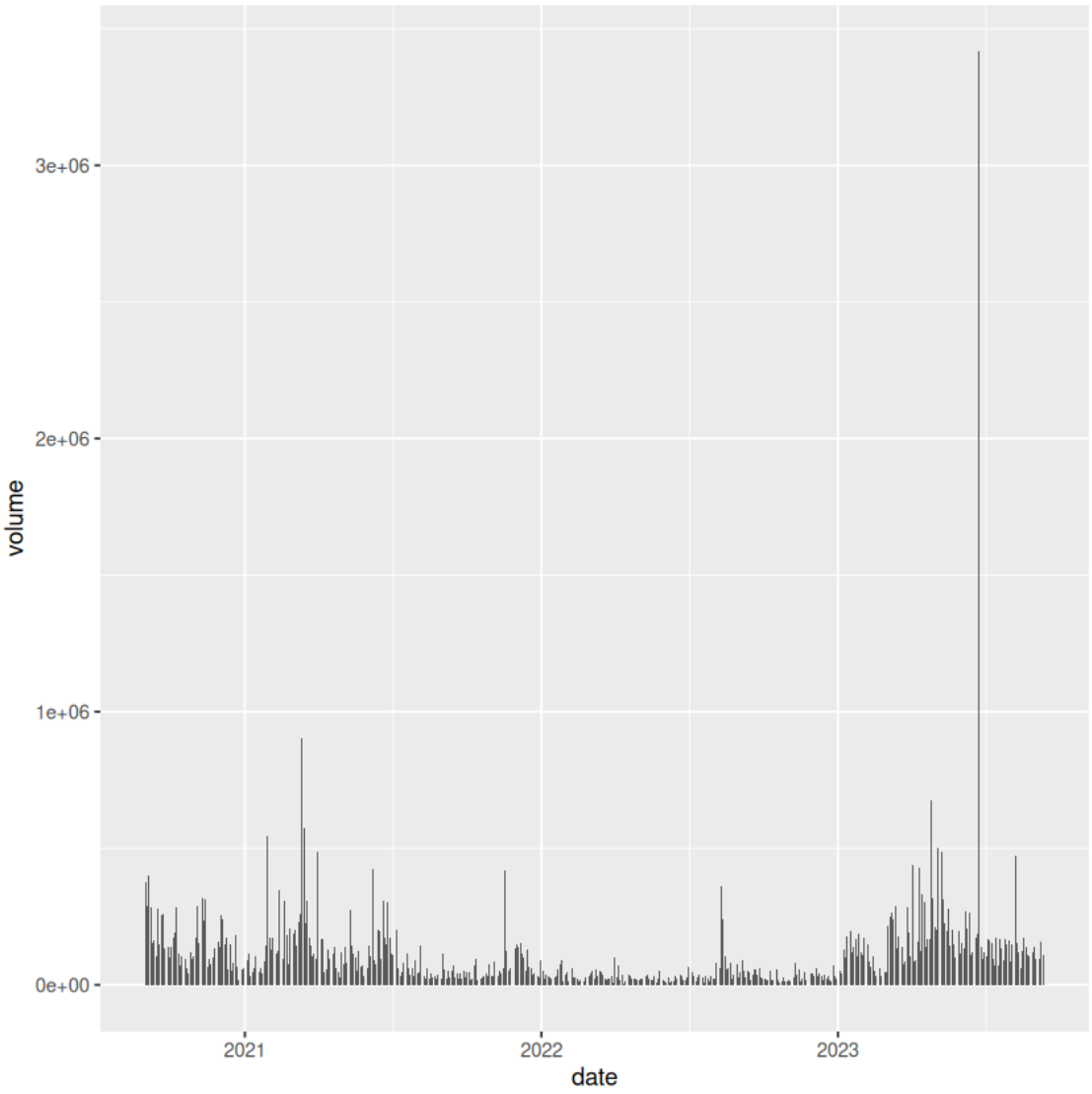
```
print(paste("Among our five tickers, ", ticker_largest_d_return, "is the one␣
 ↪with the largest daily return."))

# display the stock price chart for Potbelly
df_PBPB_prccd = data.frame(
   date = stock_data_PBPB$datadate,
   closing_price = stock_data_PBPB$prccd
   )

ggplot(df_PBPB_prccd, aes(x=date, y=closing_price)) +
   geom_line(color = "blue")

# display the stock price chart in terms of highs and lows for Potbelly
df_PBPB_highs = data.frame(
   date = stock_data_PBPB$datadate,
   closing_price = stock_data_PBPB$prchd,
   graph = "Highs"
   )

df_PBPB_highs <- subset(df_PBPB_highs, format(date, "%Y") == "2021")

df_PBPB_lows = data.frame(
   date = stock_data_PBPB$datadate,
   closing_price = stock_data_PBPB$prcld,
   graph = "Lows"
   )

df_PBPB_lows <- subset(df_PBPB_lows, format(date, "%Y") == "2021")

df_PBPB_high_low = rbind(df_PBPB_highs, df_PBPB_lows)
ggplot(df_PBPB_high_low, aes(x=date, y=closing_price, color=graph)) +
   geom_line() +
   scale_color_manual(values=c("Highs"="blue", "Lows"="red"))

# display the trading volume as a bar chart for Potbelly
df_PBPB_volume = data.frame(
   date = stock_data_PBPB$datadate,
   volume = stock_data_PBPB$cshtrd
   )

ggplot(df_PBPB_volume, aes(x=date, y=volume)) +
  geom_bar(stat = "identity", width=1)

# attempt crafting a scatter plot for Potbelly
```

PBPB 17.5862068965517 **SBUX** 9.83452172743173 **CMG** 14.7041620139316 **DPZ** 14.5523736632864

```
[1] "Among our five tickers,  PBPB is the one with the largest daily return."
```




9
8
7
6
5
closing_price
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
date
graph
Highs
Lows


3e+06
2e+06
1e+06
0e+00
volume
2021
2022
2023
date