STAT 472 Project Workflow

Adam Kiehl

5/10/2021

Set-Up and Data Wrangling

Import and style scraped data frame of daily S&P 500 prices.

```
sp_daily_highs = read.csv('./data/sp_daily_highs_lg.csv') %>%
  select(-X)
times = names(sp_daily_highs)[4:length(names(sp_daily_highs))]
times = gsub('X', '', times)
times = gsub('\\.', '-', times)
times = as.Date(times)
col_names = c('Symbol', 'Name', 'Sector', as.character(times))
names(sp_daily_highs) = col_names
```

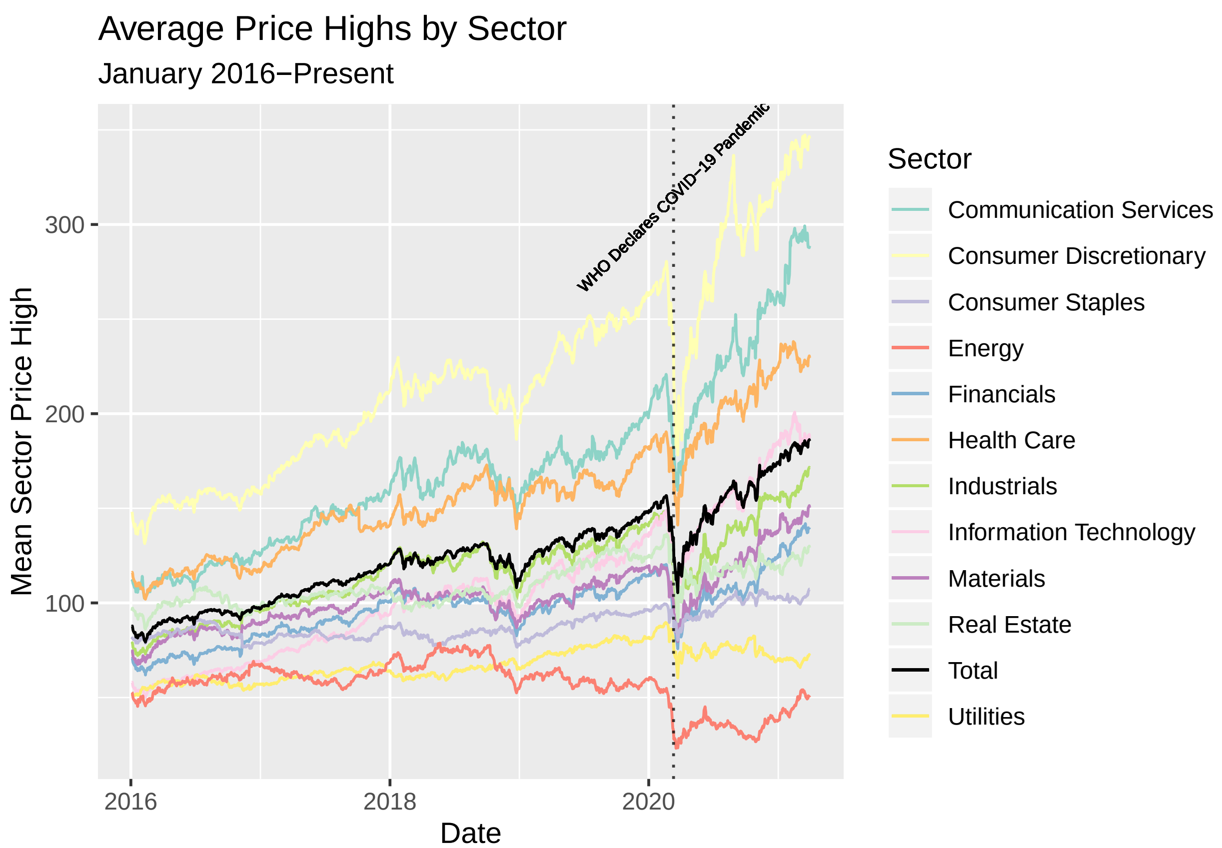
Aggregate price high data by sector and calculate daily averages. mean__tot is calculated as daily average of entire S&P 500. mean__groupa, mean__groupb, and mean__groupc are also calculated and the justification for those groupings is given later on.

```
high_avgs = data.frame(date = times)
ind = sp_daily_highs %>%
  filter(Sector == 'Industrials')
health = sp_daily_highs %>%
  filter(Sector == 'Health Care')
info = sp_daily_highs %>%
  filter(Sector == 'Information Technology')
comm = sp_daily_highs %>%
  filter(Sector == 'Communication Services')
con = sp_daily_highs %>%
  filter(Sector == 'Consumer Discretionary')
ut = sp_daily_highs %>%
  filter(Sector == 'Utilities')
fin = sp_daily_highs %>%
  filter(Sector == 'Financials')
mat = sp_daily_highs %>%
  filter(Sector == 'Materials')
rlest = sp_daily_highs %>%
  filter(Sector == 'Real Estate')
constpl = sp_daily_highs %>%
  filter(Sector == 'Consumer Staples')
nrg = sp_daily_highs %>%
  filter(Sector == 'Energy')
```

```r
groupa = rbind(comm, con, info)
groupb = rbind(fin, ind, mat)
groupc = rbind(rlest, ut)
mean_ind = c()
mean_health = c()
mean_info = c()
mean_comm = c()
mean_con = c()
mean_ut = c()
mean_fin = c()
mean_mat = c()
mean_rlest = c()
mean_constpl = c()
mean_nrg = c()
mean_tot = c()
mean_groupa = c()
mean_groupb = c()
mean_groupc = c()
for (i in 4:(3+length(times))) {
  mean_ind = c(mean_ind, as.numeric(sapply(ind[i], mean)))
  mean_health = c(mean_health, as.numeric(sapply(health[i], mean)))
  mean_info = c(mean_info, as.numeric(sapply(info[i], mean)))
  mean_comm = c(mean_comm, as.numeric(sapply(comm[i], mean)))
  mean_con = c(mean_con, as.numeric(sapply(con[i], mean)))
  mean_ut = c(mean_ut, as.numeric(sapply(ut[i], mean)))
  mean_fin = c(mean_fin, as.numeric(sapply(fin[i], mean)))
  mean_mat = c(mean_mat, as.numeric(sapply(mat[i], mean)))
  mean_rlest = c(mean_rlest, as.numeric(sapply(rlest[i], mean)))
  mean_constpl = c(mean_constpl, as.numeric(sapply(constpl[i], mean)))
  mean_nrg = c(mean_nrg, as.numeric(sapply(nrg[i], mean)))
  mean_tot = c(mean_tot, as.numeric(sapply(sp_daily_highs[i], mean)))
  mean_groupa = c(mean_groupa, as.numeric(sapply(groupa[i], mean)))
  mean_groupb = c(mean_groupb, as.numeric(sapply(groupb[i], mean)))
  mean_groupc = c(mean_groupc, as.numeric(sapply(groupc[i], mean)))
}
high_avgs$mean_ind = mean_ind
high_avgs$mean_health = mean_health
high_avgs$mean_info = mean_info
high_avgs$mean_comm = mean_comm
high_avgs$mean_con = mean_con
high_avgs$mean_ut = mean_ut
high_avgs$mean_fin = mean_fin
high_avgs$mean_mat = mean_mat
high_avgs$mean_rlest = mean_rlest
high_avgs$mean_constpl = mean_constpl
high_avgs$mean_nrg = mean_nrg
high_avgs$mean_tot = mean_tot
high_avgs$groupa = mean_groupa
high_avgs$groupb = mean_groupb
high_avgs$groupc = mean_groupc
```

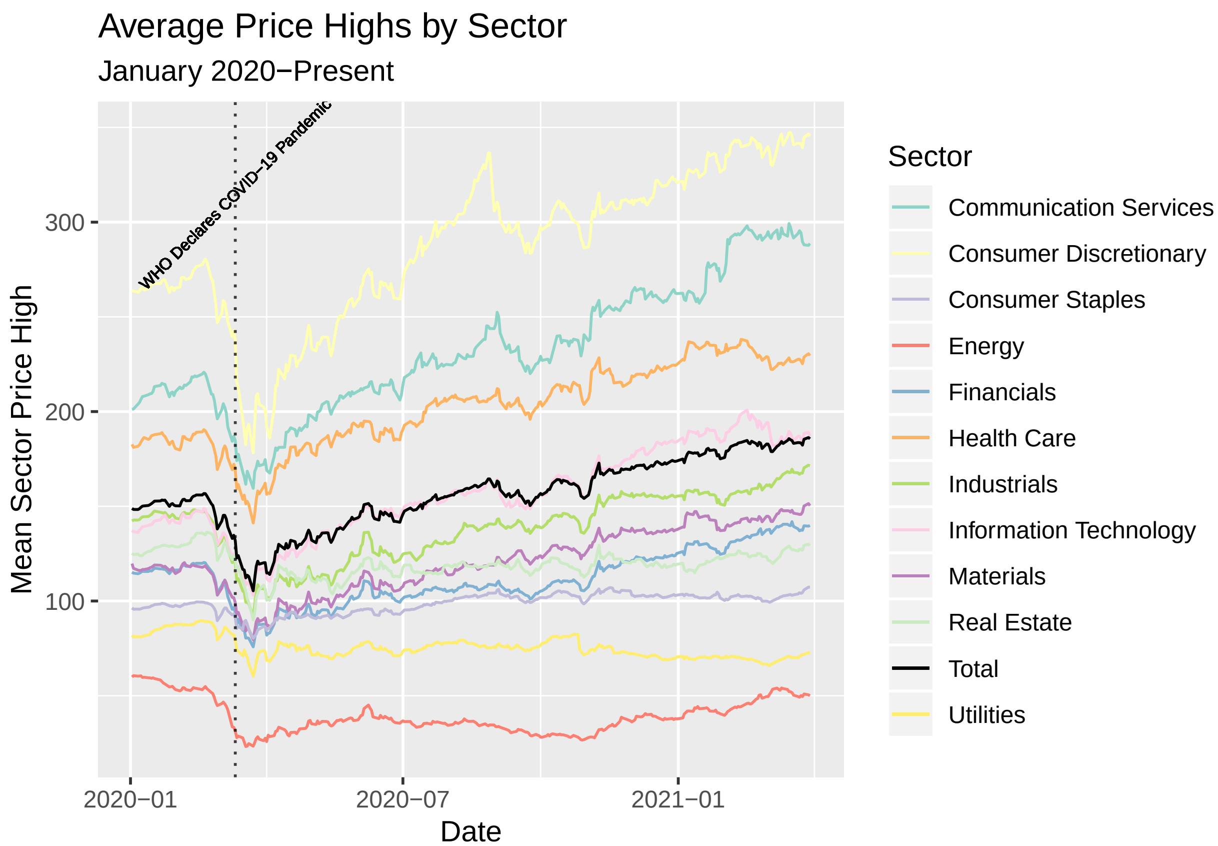
Plot daily average price highs by sector.

```
ggplot(high_avgs) +
  geom_line(aes(x = date, y = mean_ind, color = 'Industrials')) +
  geom_line(aes(x = date, y = mean_info, color = 'Information Technology')) +
  geom_line(aes(x = date, y = mean_comm, color = 'Communication Services')) +
  geom_line(aes(x = date, y = mean_con, color = 'Consumer Discretionary')) +
  geom_line(aes(x = date, y = mean_ut, color = 'Utilities')) +
  geom_line(aes(x = date, y = mean_fin, color = 'Financials')) +
  geom_line(aes(x = date, y = mean_mat, color = 'Materials')) +
  geom_line(aes(x = date, y = mean_rlest, color = 'Real Estate')) +
  geom_line(aes(x = date, y = mean_constpl, color = 'Consumer Staples')) +
  geom_line(aes(x = date, y = mean_nrg, color = 'Energy')) +
  geom_line(aes(x = date, y = mean_health, color = 'Health Care')) +
  geom_line(aes(x = date, y = mean_tot, color = 'Total')) +
  geom_vline(xintercept = as.Date('2020-03-11'), linetype = 3, alpha = .75) +
  geom_text(aes(x = as.Date('2020-03-11'), y = 315, label = 'WHO Declares COVID-19 Pandemic'), angle = 4
  scale_color_manual(values = c(
    'Communication Services' = brewer.pal(12, 'Set3')[1],
    'Consumer Discretionary' = brewer.pal(12, 'Set3')[2],
    'Consumer Staples' = brewer.pal(12, 'Set3')[3],
    'Energy' = brewer.pal(12, 'Set3')[4],
    'Financials' = brewer.pal(12, 'Set3')[5],
    'Health Care' = brewer.pal(12, 'Set3')[6],
    'Industrials' = brewer.pal(12, 'Set3')[7],
    'Information Technology' = brewer.pal(12, 'Set3')[8],
    'Materials' = brewer.pal(12, 'Set3')[10],
    'Real Estate' = brewer.pal(12, 'Set3')[11],
    'Total'= 'black',
    'Utilities' = brewer.pal(12, 'Set3')[12]
  )) +
  labs(x='Date', y='Mean Sector Price High', title='Average Price Highs by Sector', subtitle='January 20
```




```r
ggplot(high_avgs %>% filter(date > '2020-01-01')) +
  geom_line(aes(x = date, y = mean_ind, color = 'Industrials')) +
  geom_line(aes(x = date, y = mean_info, color = 'Information Technology')) +
  geom_line(aes(x = date, y = mean_comm, color = 'Communication Services')) +
  geom_line(aes(x = date, y = mean_con, color = 'Consumer Discretionary')) +
  geom_line(aes(x = date, y = mean_ut, color = 'Utilities')) +
  geom_line(aes(x = date, y = mean_fin, color = 'Financials')) +
  geom_line(aes(x = date, y = mean_mat, color = 'Materials')) +
  geom_line(aes(x = date, y = mean_rlest, color = 'Real Estate')) +
  geom_line(aes(x = date, y = mean_constpl, color = 'Consumer Staples')) +
  geom_line(aes(x = date, y = mean_nrg, color = 'Energy')) +
  geom_line(aes(x = date, y = mean_health, color = 'Health Care')) +
  geom_line(aes(x = date, y = mean_tot, color = 'Total')) +
  geom_vline(xintercept = as.Date('2020-03-11'), linetype = 3, alpha = .75) +
  geom_text(aes(x = as.Date('2020-03-11'), y = 315, label = 'WHO Declares COVID-19 Pandemic'), angle = 
  scale_color_manual(values = c(
    'Communication Services' = brewer.pal(12, 'Set3')[1],
    'Consumer Discretionary' = brewer.pal(12, 'Set3')[2],
    'Consumer Staples' = brewer.pal(12, 'Set3')[3],
    'Energy' = brewer.pal(12, 'Set3')[4],
    'Financials' = brewer.pal(12, 'Set3')[5],
    'Health Care' = brewer.pal(12, 'Set3')[6],
    'Industrials' = brewer.pal(12, 'Set3')[7],
    'Information Technology' = brewer.pal(12, 'Set3')[8],
    'Materials' = brewer.pal(12, 'Set3')[10],
    'Real Estate' = brewer.pal(12, 'Set3')[11],
```

```
    'Total'= 'black',
    'Utilities' = brewer.pal(12, 'Set3')[12]
  )) +
  labs(x='Date', y='Mean Sector Price High', title='Average Price Highs by Sector', subtitle='January 20
```



Write data frame to .csv for safe-keeping.

```
write.csv(high_avgs, './data/high_avgs.csv')
```

Separate data into training and testing sets. Data before 2021 will be used for training.

```
high_train = high_avgs %>%
  filter(as.Date(date) < as.Date('2021-01-01'))
high_test = high_avgs %>%
  filter(as.Date(date) >= as.Date('2021-01-01'))
```

Structural Breaks

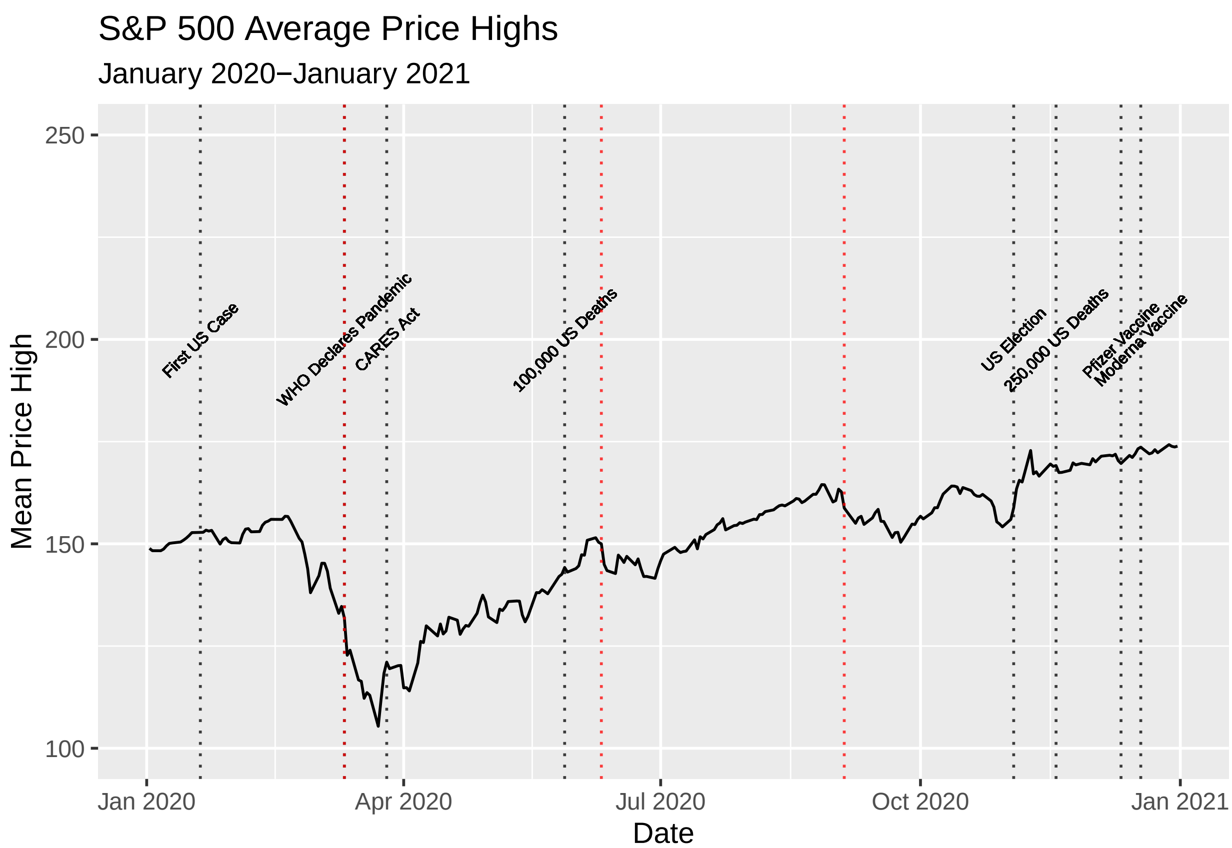
Identify strucutral breaks within 2020 data using Chow Test in strucchange package.

```
high_2020 = high_train %>%
  filter(as.Date(date) > '2020-01-01')
print(high_2020$date[breakpoints(high_2020$mean_tot ~ high_2020$date)$breakpoints])
```

```
## [1] "2020-03-11" "2020-06-10" "2020-09-04"
```

Plot the time series to visualize structural breakpoints. The first break aligns closely with the WHO declaring the COVID-19 pandemic and the initial March shutdown. The second break corresponds to the week where many of the initial stay-at-home orders expired. The third break corresponds roughly with the start of the school and the onset of fall.

```
plt <- ggplot(high_2020) +
  geom_line(aes(x = date, y = mean_tot)) +
  geom_vline(xintercept = as.Date('2020-01-20'), linetype = 3, alpha = .75) +
  geom_text(aes(x = as.Date('2020-01-20'), y = 200, label = 'First US Case'), angle = 45, size = 2) +
  geom_vline(xintercept = as.Date('2020-03-11'), linetype = 3, alpha = .75) +
  geom_text(aes(x = as.Date('2020-03-11'), y = 200, label = 'WHO Declares Pandemic'), angle = 45, size =
  geom_vline(xintercept = as.Date('2020-03-26'), linetype = 3, alpha = .75) +
  geom_text(aes(x = as.Date('2020-03-26'), y = 200, label = 'CARES Act'), angle = 45, size = 2) +
  geom_vline(xintercept = as.Date('2020-05-28'), linetype = 3, alpha = .75) +
  geom_text(aes(x = as.Date('2020-05-28'), y = 200, label = '100,000 US Deaths'), angle = 45, size = 2)
  geom_vline(xintercept = as.Date('2020-11-03'), linetype = 3, alpha = .75) +
  geom_text(aes(x = as.Date('2020-11-03'), y = 200, label = 'US Election'), angle = 45, size = 2) +
  geom_vline(xintercept = as.Date('2020-11-18'), linetype = 3, alpha = .75) +
  geom_text(aes(x = as.Date('2020-11-18'), y = 200, label = '250,000 US Deaths'), angle = 45, size = 2)
  geom_vline(xintercept = as.Date('2020-12-11'), linetype = 3, alpha = .75) +
  geom_text(aes(x = as.Date('2020-12-11'), y = 200, label = 'Pfizer Vaccine'), angle = 45, size = 2) +
  geom_vline(xintercept = as.Date('2020-12-18'), linetype = 3, alpha = .75) +
  geom_text(aes(x = as.Date('2020-12-18'), y = 200, label = 'Moderna Vaccine'), angle = 45, size = 2) +
  labs(x='Date', y='Mean Price High', title='S&P 500 Average Price Highs', subtitle='January 2020-Janua
  ylim(100, 250)
for (brk in breakpoints(high_2020$mean_tot ~ high_2020$date)$breakpoints) {
  plt <- plt + geom_vline(xintercept = as.Date(high_2020$date[brk]), linetype = 3, alpha = .75, color =
}

plt
```



Characterize state periods as defined by structural breaks.

```
prds_high = c(rep(1, 47), rep(2, 63), rep(3, 61), rep(4, 82))
high_2020 %>%
  mutate(period = prds_high) %>%
  group_by(period) %>%
  summarize(Mean = mean(mean_tot), Var = var(mean_tot), .groups = 'keep')
```

```
## # A tibble: 4 x 3
## # Groups:   period [4]
##   period  Mean   Var
##    <dbl> <dbl> <dbl>
## 1      1  150.  29.7
## 2      2  130. 117.
## 3      3  153.  45.5
## 4      4  164.  49.7
```

Generate dummy variables and interactions for modelling break periods and seasons.

```
high_train$period1 = with(rle(high_train$date < '2020-03-11'), rep(as.integer(values), lengths))
high_train$period2 = with(rle((high_train$date >= '2020-03-11') & (high_train$date < '2020-06-10')), rep
high_train$period3 = with(rle((high_train$date >= '2020-06-10') & (high_train$date < '2020-09-04')), rep
high_train$period4 = with(rle(high_train$date >= '2020-09-04'), rep(as.integer(values), lengths))

high_train$interConstpl2 = high_train$mean_constpl * high_train$period2
high_train$interConstpl3 = high_train$mean_constpl * high_train$period3
```

```
high_train$interComm3 = high_train$mean_comm * high_train$period3

spring = c()
summer = c()
fall = c()
winter = c()
for (i in 1:length(high_train$date)) {
  mon = as.numeric(format(high_train$date[i], '%m'))
  if (mon %in% c(3, 4, 5)) {
    spring = c(spring, 1)
    summer = c(summer, 0)
    fall = c(fall, 0)
    winter = c(winter, 0)
  }
  else if (mon %in% c(6, 7, 8)) {
    spring = c(spring, 0)
    summer = c(summer, 1)
    fall = c(fall, 0)
    winter = c(winter, 0)
  }
  else if (mon %in% c(9, 10, 11)) {
    spring = c(spring, 0)
    summer = c(summer, 0)
    fall = c(fall, 1)
    winter = c(winter, 0)
  }
  else {
    spring = c(spring, 0)
    summer = c(summer, 0)
    fall = c(fall, 0)
    winter = c(winter, 1)
  }
}
```

Pre-Modelling

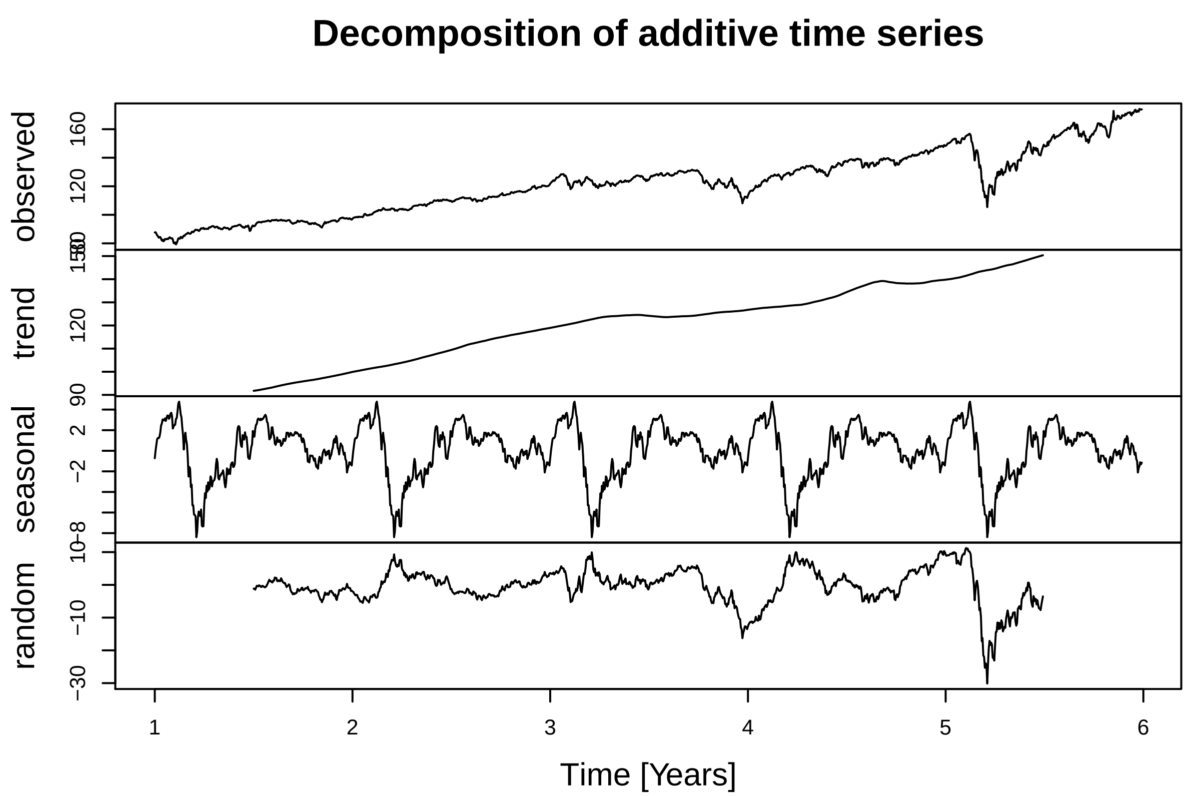
Create time series from sector averages with 252 trading days in a year.

```
high_ts = data.frame(date = as.Date(high_train$date))
high_ts$high_ind_ts = ts(high_train$mean_ind, frequency = 252)
high_ts$high_health_ts = ts(high_train$mean_health, frequency = 252)
high_ts$high_info_ts = ts(high_train$mean_info, frequency = 252)
high_ts$high_comm_ts = ts(high_train$mean_comm, frequency = 252)
high_ts$high_con_ts = ts(high_train$mean_con, frequency = 252)
high_ts$high_ut_ts = ts(high_train$mean_ut, frequency = 252)
high_ts$high_fin_ts = ts(high_train$mean_fin, frequency = 252)
high_ts$high_mat_ts = ts(high_train$mean_mat, frequency = 252)
high_ts$high_rlest_ts = ts(high_train$mean_rlest, frequency = 252)
high_ts$high_constpl_ts = ts(high_train$mean_constpl, frequency = 252)
high_ts$high_nrg_ts = ts(high_train$mean_nrg, frequency = 252)
high_ts$high_tot_ts = ts(high_train$mean_tot, frequency = 252)
high_ts$high_groupa_ts = ts(high_train$groupa, frequency = 252)
```

```
high_ts$high_groupb_ts = ts(high_train$groupb, frequency = 252)
high_ts$high_groupc_ts = ts(high_train$groupc, frequency = 252)
high_ts$period2 = ts(high_train$period2, frequency = 252)
high_ts$period3 = ts(high_train$period3, frequency = 252)
high_ts$period4 = ts(high_train$period4, frequency = 252)
high_ts$interConstpl2 = ts(high_train$interConstpl2, frequency = 252)
high_ts$interConstpl3 = ts(high_train$interConstpl3, frequency = 252)
high_ts$interComm3 = ts(high_train$interComm3, frequency = 252)
high_ts$spring = ts(spring, frequency = 252)
high_ts$summer = ts(summer, frequency = 252)
high_ts$fall = ts(fall, frequency = 252)
```

These time series contain both seasonal and linear trends that must be removed to achieve stationarity.

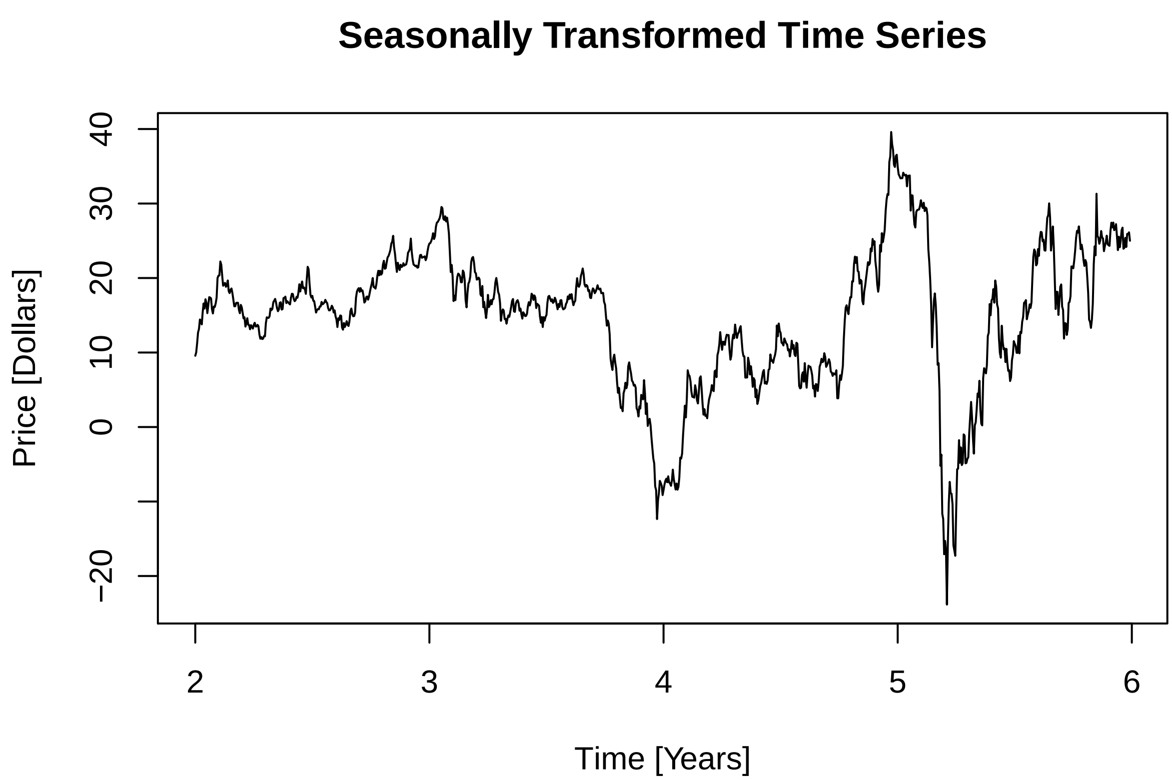
```
plot(decompose(high_ts$high_tot_ts), xlab = 'Time [Years]')
```



Apply seasonal differencing transform across all time series to remove seasonality. An example plot is shown.

```
high_tr1 = data.frame(date = as.Date(high_train$date[253:1259]))
high_tr1$high_ind_tr = diff(high_ts$high_ind_ts, lag = 252, differences = 1)
high_tr1$high_health_tr = diff(high_ts$high_health_ts, lag = 252, differences = 1)
high_tr1$high_info_tr = diff(high_ts$high_info_ts, lag = 252, differences = 1)
high_tr1$high_comm_tr = diff(high_ts$high_comm_ts, lag = 252, differences = 1)
high_tr1$high_con_tr = diff(high_ts$high_con_ts, lag = 252, differences = 1)
```

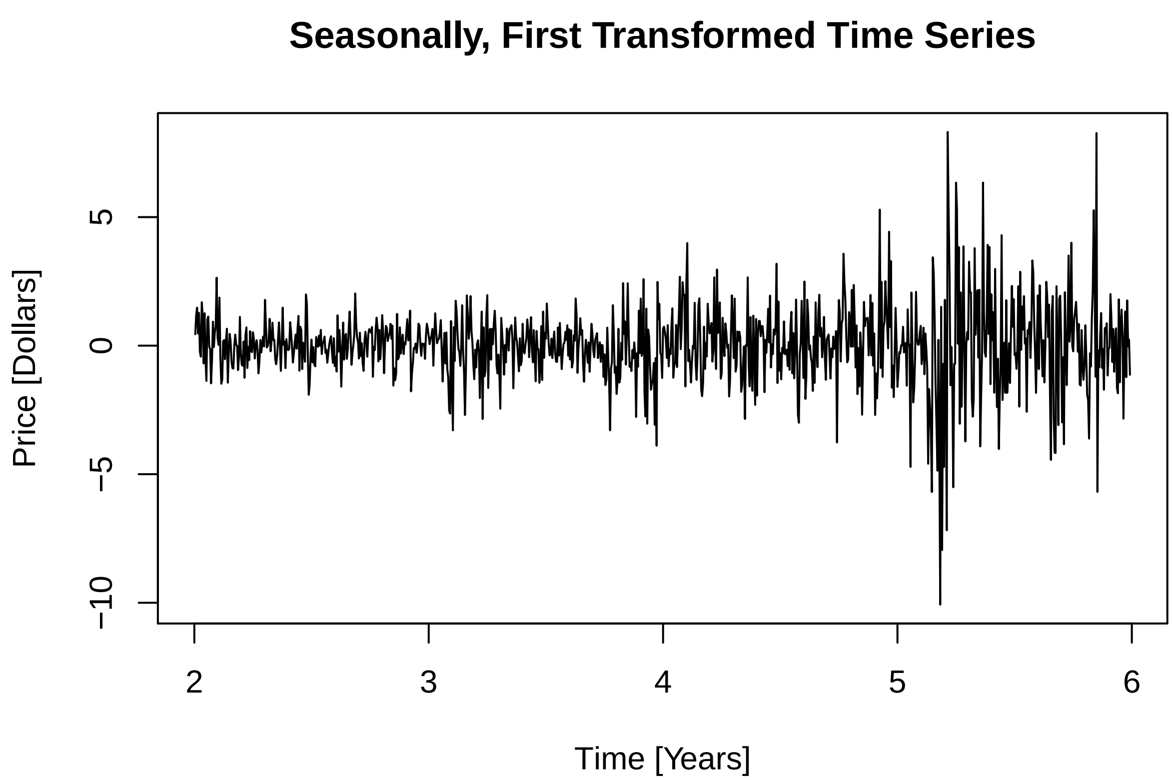
```
high_tr1$high_ut_tr = diff(high_ts$high_ut_ts, lag = 252, differences = 1)
high_tr1$high_fin_tr = diff(high_ts$high_fin_ts, lag = 252, differences = 1)
high_tr1$high_mat_tr = diff(high_ts$high_mat_ts, lag = 252, differences = 1)
high_tr1$high_rlest_tr = diff(high_ts$high_rlest_ts, lag = 252, differences = 1)
high_tr1$high_constpl_tr = diff(high_ts$high_constpl_ts, lag = 252, differences = 1)
high_tr1$high_nrg_tr = diff(high_ts$high_nrg_ts, lag = 252, differences = 1)
high_tr1$high_tot_tr = diff(high_ts$high_tot_ts, lag = 252, differences = 1)
high_tr1$high_groupa_tr = diff(high_ts$high_groupa_ts, lag = 252, differences = 1)
high_tr1$high_groupb_tr = diff(high_ts$high_groupb_ts, lag = 252, differences = 1)
high_tr1$high_groupc_tr = diff(high_ts$high_groupc_ts, lag = 252, differences = 1)
high_tr1$period2 = diff(high_ts$period2, lag = 252, differences = 1)
high_tr1$period3 = diff(high_ts$period3, lag = 252, differences = 1)
high_tr1$period4 = diff(high_ts$period4, lag = 252, differences = 1)
high_tr1$interConstpl2 = diff(high_ts$interConstpl2, lag = 252, differences = 1)
high_tr1$interConstpl3 = diff(high_ts$interConstpl3, lag = 252, differences = 1)
high_tr1$interComm3 = diff(high_ts$interComm3, lag = 252, differences = 1)
plot(high_tr1$high_tot_tr, main='Seasonally Transformed Time Series', xlab='Time [Years]', ylab='Price
```



Apply first-differencing transform across all time series to remove trends. An example plot is shown.

```
high_tr2 = data.frame(date = as.Date(high_train$date[254:1259]))
high_tr2$high_ind_tr = diff(high_tr1$high_ind_tr, lag = 1, differences = 1)
high_tr2$high_health_tr = diff(high_tr1$high_health_tr, lag = 1, differences = 1)
high_tr2$high_info_tr = diff(high_tr1$high_info_tr, lag = 1, differences = 1)
high_tr2$high_comm_tr = diff(high_tr1$high_comm_tr, lag = 1, differences = 1)
high_tr2$high_con_tr = diff(high_tr1$high_con_tr, lag = 1, differences = 1)
```

```
high_tr2$high_ut_tr = diff(high_tr1$high_ut_tr, lag = 1, differences = 1)
high_tr2$high_fin_tr = diff(high_tr1$high_fin_tr, lag = 1, differences = 1)
high_tr2$high_mat_tr = diff(high_tr1$high_mat_tr, lag = 1, differences = 1)
high_tr2$high_rlest_tr = diff(high_tr1$high_rlest_tr, lag = 1, differences = 1)
high_tr2$high_constpl_tr = diff(high_tr1$high_constpl_tr, lag = 1, differences = 1)
high_tr2$high_nrg_tr = diff(high_tr1$high_nrg_tr, lag = 1, differences = 1)
high_tr2$high_tot_tr = diff(high_tr1$high_tot_tr, lag = 1, differences = 1)
high_tr2$high_groupa_tr = diff(high_tr1$high_groupa_tr, lag = 1, differences = 1)
high_tr2$high_groupb_tr = diff(high_tr1$high_groupb_tr, lag = 1, differences = 1)
high_tr2$high_groupc_tr = diff(high_tr1$high_groupc_tr, lag = 1, differences = 1)
high_tr2$period2 = diff(high_tr1$period2, lag = 1, differences = 1)
high_tr2$period3 = diff(high_tr1$period3, lag = 1, differences = 1)
high_tr2$period4 = diff(high_tr1$period4, lag = 1, differences = 1)
high_tr2$interConstpl2 = diff(high_tr1$interConstpl2, lag = 1, differences = 1)
high_tr2$interConstpl3 = diff(high_tr1$interConstpl3, lag = 1, differences = 1)
high_tr2$interComm3 = diff(high_tr1$interComm3, lag = 1, differences = 1)
plot(high_tr2$high_tot_tr, main='Seasonally, First Transformed Time Series', xlab='Time [Years]', ylab=
```



Generate a correlation matrix between sectors using differenced time series. From this matrix, the following groups can be estimated: groupA- comm, con, info; groupB- fin, ind, mat; groupC- rlest, ut.

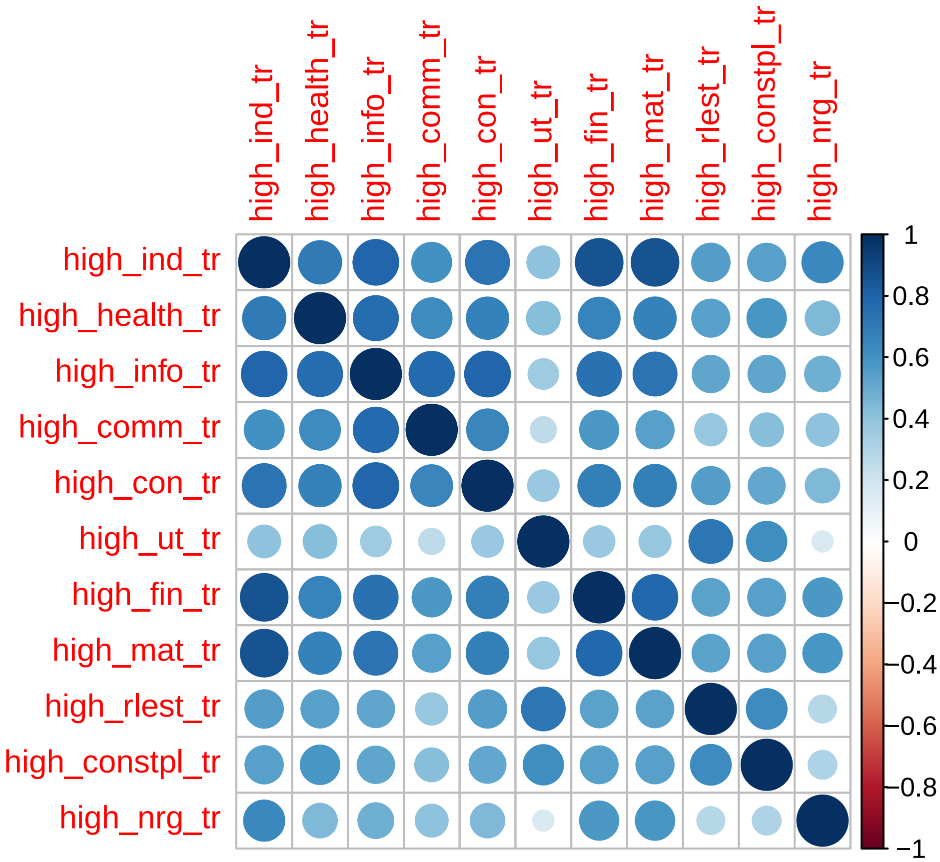
```
cor(high_tr2 %>%
      select(-c(date, high_tot_tr, high_groupa_tr, high_groupb_tr, high_groupc_tr, period2, period3, pe
      sapply(unlist),
    high_tr2 %>%
      select(-c(date, high_tot_tr, high_groupa_tr, high_groupb_tr, high_groupc_tr, period2, period3, pe
```

```
    sapply(unlist))
```

```
##                  high_ind_tr high_health_tr high_info_tr high_comm_tr
## high_ind_tr        1.0000000      0.7078052    0.7900643    0.6008612
## high_health_tr     0.7078052      1.0000000    0.7691812    0.6253192
## high_info_tr       0.7900643      0.7691812    1.0000000    0.7755363
## high_comm_tr       0.6008612      0.6253192    0.7755363    1.0000000
## high_con_tr        0.7323629      0.6759297    0.7958813    0.6594727
## high_ut_tr         0.4027815      0.4243250    0.3501842    0.2547197
## high_fin_tr        0.8604983      0.6637864    0.7491819    0.5782941
## high_mat_tr        0.8621613      0.6789429    0.7312681    0.5485360
## high_rlest_tr      0.5551211      0.5447146    0.5257525    0.3868602
## high_constpl_tr    0.5499334      0.5867397    0.5257092    0.4288759
## high_nrg_tr        0.6403182      0.4473263    0.4807309    0.4054323
##                  high_con_tr high_ut_tr high_fin_tr high_mat_tr high_rlest_tr
## high_ind_tr        0.7323629  0.4027815   0.8604983   0.8621613     0.5551211
## high_health_tr     0.6759297  0.4243250   0.6637864   0.6789429     0.5447146
## high_info_tr       0.7958813  0.3501842   0.7491819   0.7312681     0.5257525
## high_comm_tr       0.6594727  0.2547197   0.5782941   0.5485360     0.3868602
## high_con_tr        1.0000000  0.3731022   0.6847692   0.6852676     0.5520245
## high_ut_tr         0.3731022  1.0000000   0.3701925   0.3839156     0.7250220
## high_fin_tr        0.6847692  0.3701925   1.0000000   0.7838640     0.5349943
## high_mat_tr        0.6852676  0.3839156   0.7838640   1.0000000     0.5324632
## high_rlest_tr      0.5520245  0.7250220   0.5349943   0.5324632     1.0000000
## high_constpl_tr    0.5128749  0.6117133   0.5400915   0.5462733     0.6249245
## high_nrg_tr        0.4476533  0.1643571   0.5788815   0.5839168     0.2895127
##                  high_constpl_tr high_nrg_tr
## high_ind_tr            0.5499334   0.6403182
## high_health_tr         0.5867397   0.4473263
## high_info_tr           0.5257092   0.4807309
## high_comm_tr           0.4288759   0.4054323
## high_con_tr            0.5128749   0.4476533
## high_ut_tr             0.6117133   0.1643571
## high_fin_tr            0.5400915   0.5788815
## high_mat_tr            0.5462733   0.5839168
## high_rlest_tr          0.6249245   0.2895127
## high_constpl_tr        1.0000000   0.3086023
## high_nrg_tr            0.3086023   1.0000000
```

Visualize correlations between sectors.

```
corrplot(cor(high_tr2 %>%
      select(-c(date, high_tot_tr, high_groupa_tr, high_groupb_tr, high_groupc_tr, period2, period3, pe
      sapply(unlist),
    high_tr2 %>%
      select(-c(date, high_tot_tr, high_groupa_tr, high_groupb_tr, high_groupc_tr, period2, period3, pe
      sapply(unlist)))
```



Confirm groupings with exploratory factor analysis. Factor 1 is roughly equivalent to groupA. Factor 3 is roughly equivalent to groupB. Factor 2 is roughly equivalent to groupC. Factor 4 represents the energy sector, Factor 5 represents the consumer staples sector, and the health sector is here included with Factor 1.

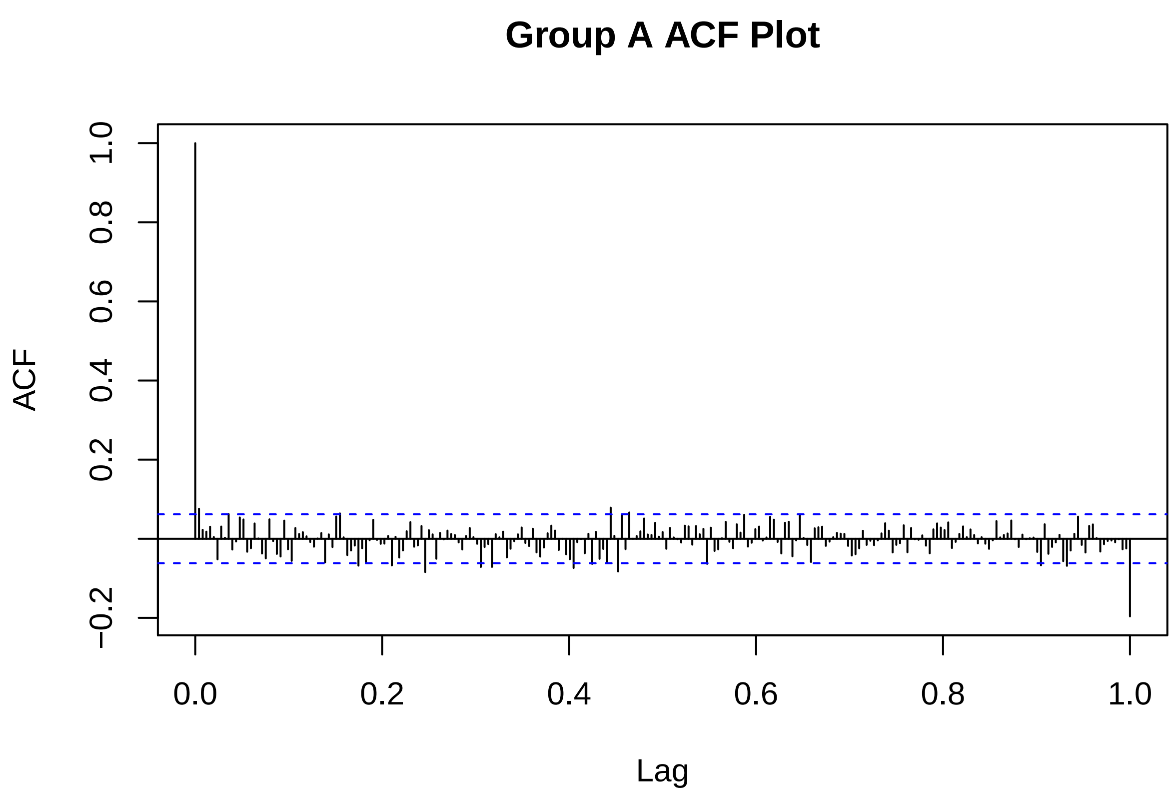
```
factanal(high_tr2[,2:12,], 6)
```

```
##
## Call:
## factanal(x = high_tr2[, 2:12, ], factors = 6)
##
## Uniquenesses:
##     high_ind_tr  high_health_tr    high_info_tr    high_comm_tr     high_con_tr
##           0.087           0.315           0.074           0.324           0.293
##      high_ut_tr      high_fin_tr      high_mat_tr   high_rlest_tr high_constpl_tr
##           0.388           0.005           0.177           0.005           0.189
##     high_nrg_tr
##           0.005
##
## Loadings:
##                Factor1 Factor2 Factor3 Factor4 Factor5 Factor6
## high_ind_tr     0.552   0.271   0.533   0.357   0.119   0.329
## high_health_tr  0.649   0.317   0.245   0.188   0.224   0.129
## high_info_tr    0.869   0.220   0.276   0.172
## high_comm_tr    0.774   0.133   0.142   0.162   0.109
## high_con_tr     0.695   0.300   0.277   0.178           0.137
## high_ut_tr      0.149   0.716   0.106           0.244
```

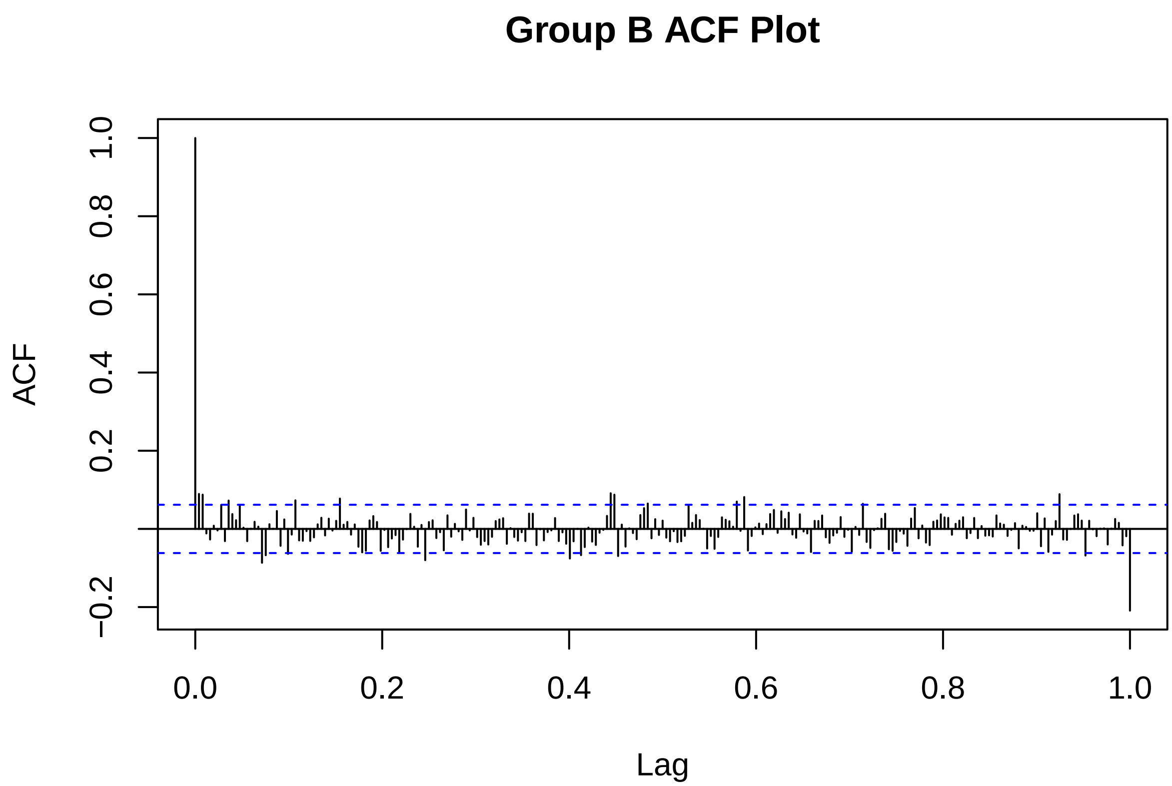
```
## high_fin_tr      0.486   0.245   0.779   0.274   0.131
## high_mat_tr      0.503   0.278   0.472   0.319   0.149   0.383
## high_rlest_tr    0.290   0.929   0.181   0.112
## high_constpl_tr  0.320   0.545   0.183   0.111   0.603
## high_nrg_tr      0.271           0.215   0.929
##
##                 Factor1 Factor2 Factor3 Factor4 Factor5 Factor6
## SS loadings       3.335   2.147   1.471   1.316   0.557   0.312
## Proportion Var    0.303   0.195   0.134   0.120   0.051   0.028
## Cumulative Var    0.303   0.498   0.632   0.752   0.802   0.831
##
## Test of the hypothesis that 6 factors are sufficient.
## The chi square statistic is 8.4 on 4 degrees of freedom.
## The p-value is 0.0778
```

Test time series for stability with Auto-Correlation Function plots using the Box-Jenkins method.

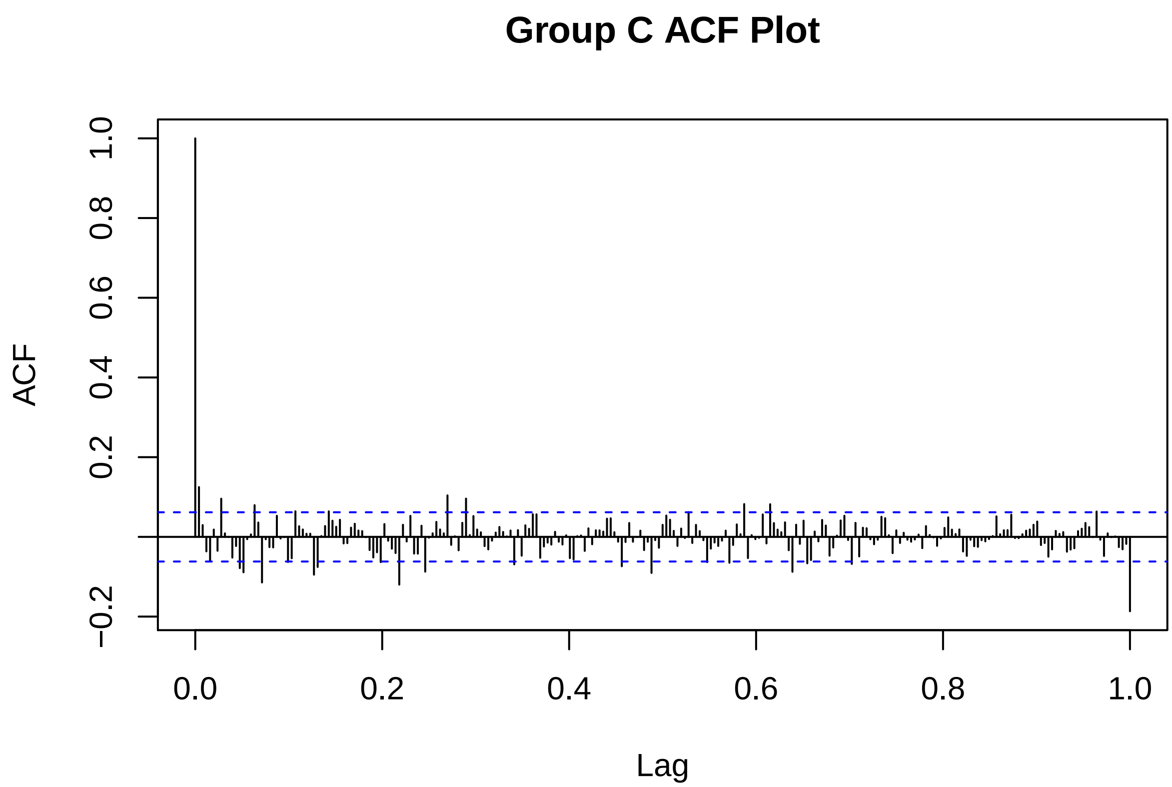
```
acf(high_tr2$high_groupa_tr, lag = 252, main='Group A ACF Plot')
```



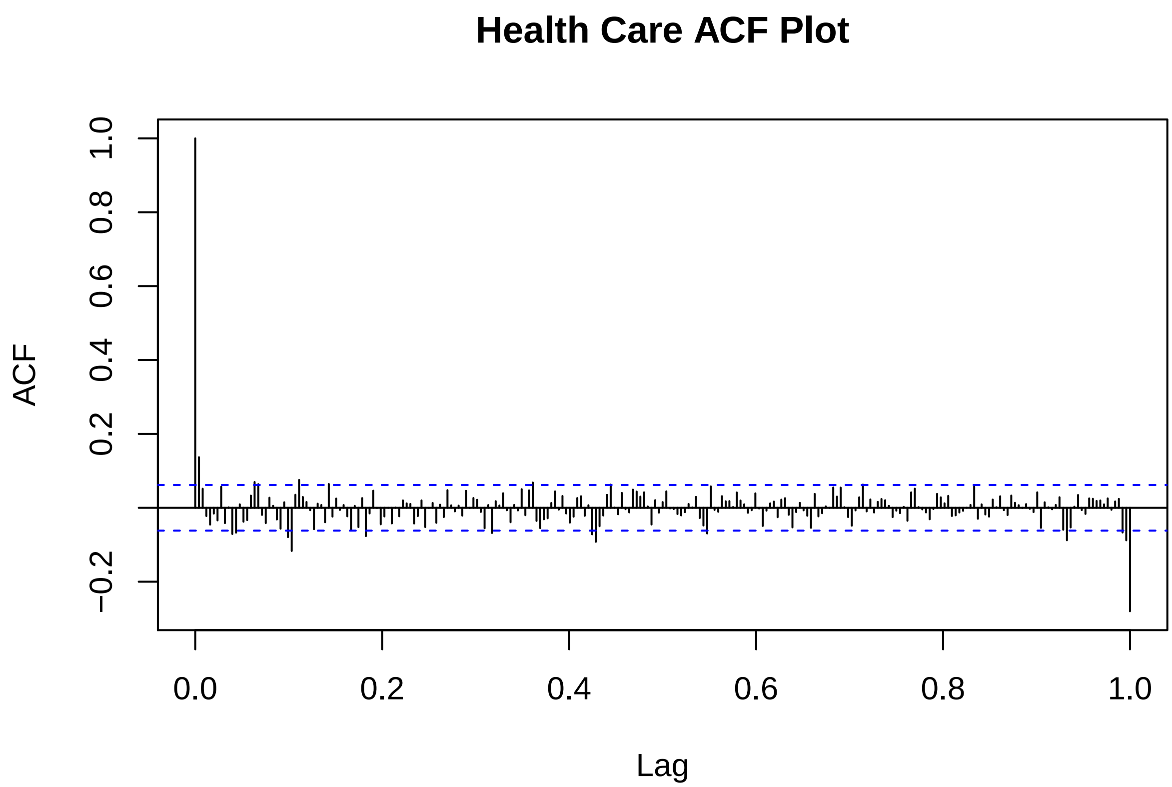
```
acf(high_tr2$high_groupb_tr, lag = 252, main='Group B ACF Plot')
```



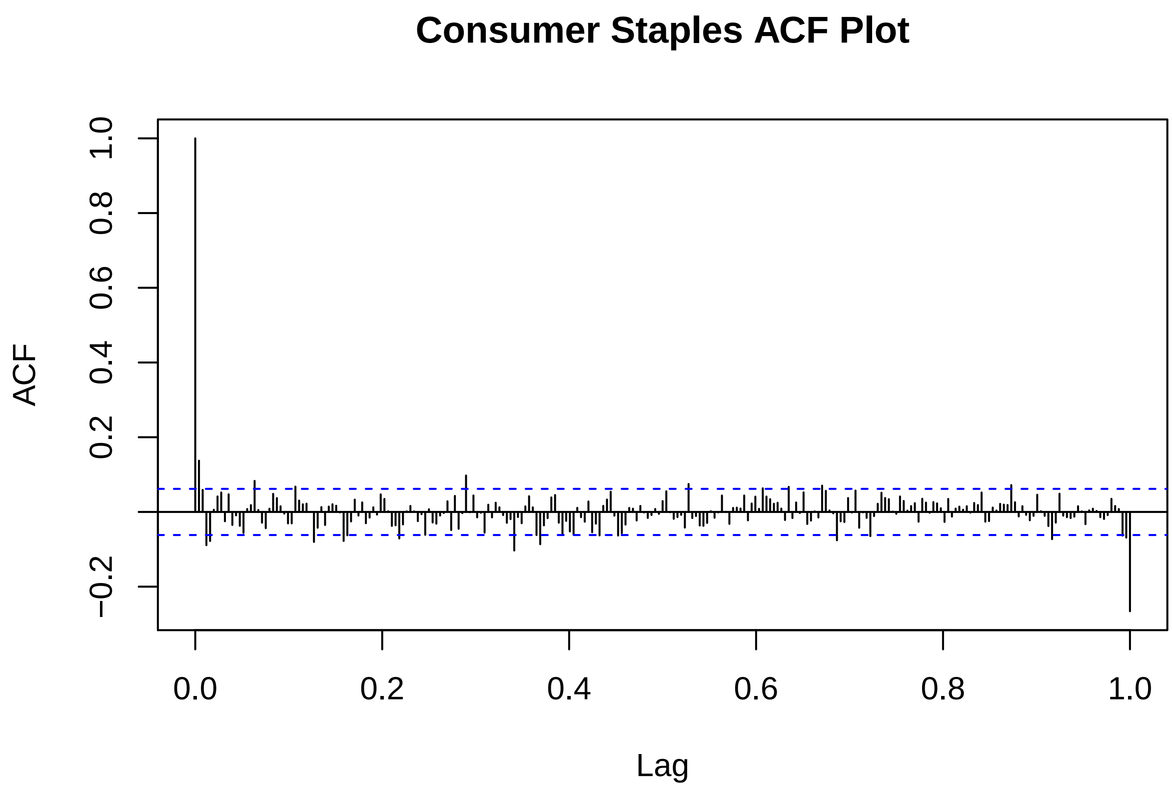

```
acf(high_tr2$high_groupc_tr, lag = 252, main='Group C ACF Plot')
```



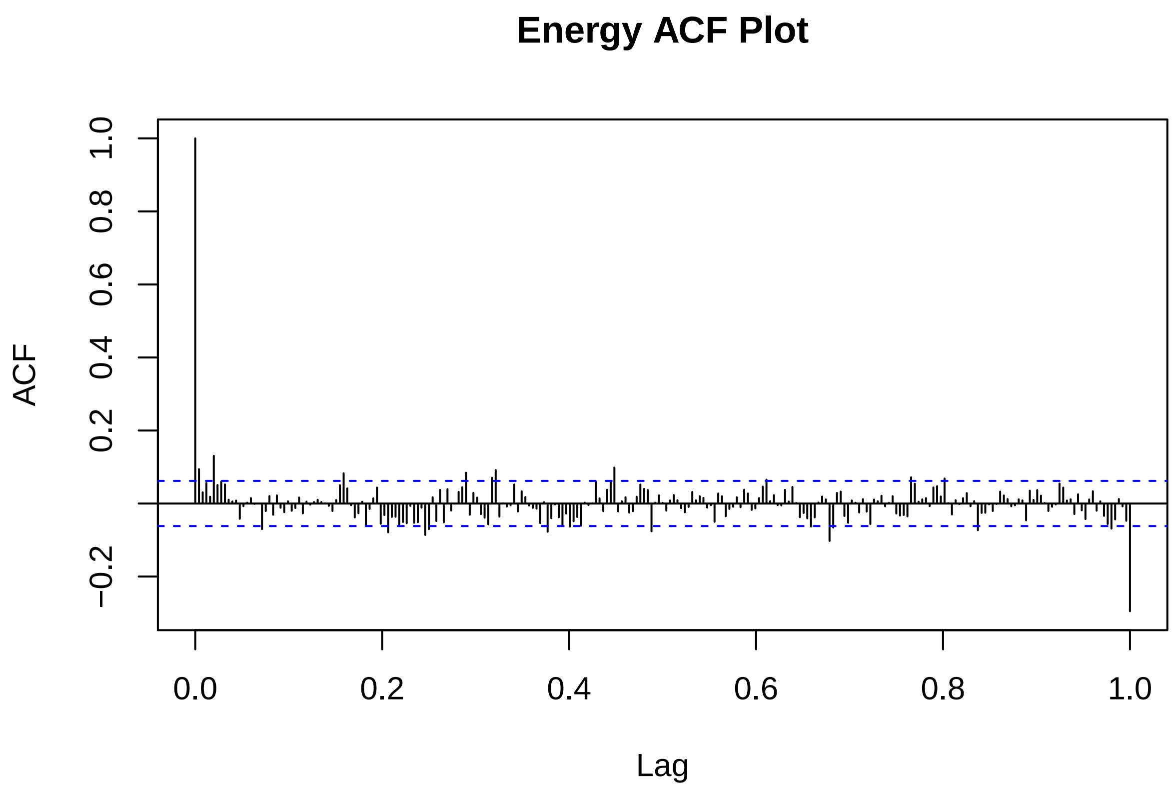

```
acf(high_tr2$high_health_tr, lag = 252, main='Health Care ACF Plot')
```




```
acf(high_tr2$high_constpl_tr, lag = 252, main='Consumer Staples ACF Plot')
```




```
acf(high_tr2$high_nrg_tr, lag = 252, main='Energy ACF Plot')
```



Modelling

Optimize AIC over ARMA paramaters p and q to tune order() with 0 differences built in.

```
final.aic = Inf
final.order = c(0, 0, 0)
for (p in 0:10) {
  for (q in 0:10) {
    tryCatch(
      {
        current.aic = AIC(Arima(high_tr2$high_tot_tr, order = c(p, 0, q), xreg = as.matrix(high_ts$peri
        if (current.aic < final.aic) {
          final.aic = current.aic
          final.order = c(p, 0, q)
        }
      },
      error = function(cond) {
      }
    )
  }
}
final.order
```

```
## [1]  7  0 10
```

Fit ARMA models for each sector grouping with dummy variable for the break period.

```
order = final.order
modelA1 = Arima(high_tr2$high_groupa_tr,
               order = order,
               xreg = as.matrix(high_ts$period2[254:1259]))
modelB1 = Arima(high_tr2$high_groupb_tr,
               order = order,
               xreg = as.matrix(high_ts$period2[254:1259]))
modelC1 = Arima(high_tr2$high_groupc_tr,
               order = order,
               xreg = as.matrix(high_ts$period2[254:1259]))
modelHealth1 = Arima(high_tr2$high_health_tr,
                    order = order,
                    xreg = as.matrix(high_ts$period2[254:1259]))
modelConstpl1 = Arima(high_tr2$high_constpl_tr,
                    order = order,
                    xreg = as.matrix(high_ts$period2[254:1259]))
modelNrg1 = Arima(high_tr2$high_nrg_tr,
                    order = order,
                    xreg = as.matrix(high_ts$period2[254:1259]))
```

Fit ARMA models for each sector grouping with dummy variables for the break period and seasons.

```
order = final.order
modelA2 = Arima(diff(high_ts$high_groupa_ts, lag = 1, differences = 1),
               order = order,
               xreg = as.matrix(cbind(high_ts$period2[-1], high_ts$spring[-1], high_ts$summer[-1], high_
modelB2 = Arima(diff(high_ts$high_groupb_ts, lag = 1, differences = 1),
               order = order,
               xreg = as.matrix(cbind(high_ts$period2[-1], high_ts$spring[-1], high_ts$summer[-1], high_
modelC2 = Arima(diff(high_ts$high_groupc_ts, lag = 1, differences = 1),
               order = order,
               xreg = as.matrix(cbind(high_ts$period2[-1], high_ts$spring[-1], high_ts$summer[-1], high_
modelHealth2 = Arima(diff(high_ts$high_health_ts, lag = 1, differences = 1),
                    order = order,
                    xreg = as.matrix(cbind(high_ts$period2[-1], high_ts$spring[-1], high_ts$summer[-1],
modelConstpl2 = Arima(diff(high_ts$high_constpl_ts, lag = 1, differences = 1),
                    order = order,
                    xreg = as.matrix(cbind(high_ts$period2[-1], high_ts$spring[-1], high_ts$summer[-1],
modelNrg2 = Arima(diff(high_ts$high_nrg_ts, lag = 1, differences = 1),
                    order = order,
                    xreg = as.matrix(cbind(high_ts$period2[-1], high_ts$spring[-1], high_ts$summer[-1],
```

Again, optimize AIC over ARMA paramaters p and q to tune order() with two differences built in.

```
final.aic = Inf
final.order = c(0, 0, 0)
for (p in 0:10) {
  for (q in 0:10) {
    tryCatch(
      {
        current.aic = AIC(Arima(high_ts$high_tot_ts, order = c(p, 2, q), xreg = high_ts$period2))
```

```
        if (current.aic < final.aic) {
          final.aic = current.aic
          final.order = c(p, 2, q)
        }
      },
      error = function(cond) {
      }
    )
  }
}

final.order
```

```
## [1] 9 2 3
```

Fit ARMA model for total time series using second-order differencing model.

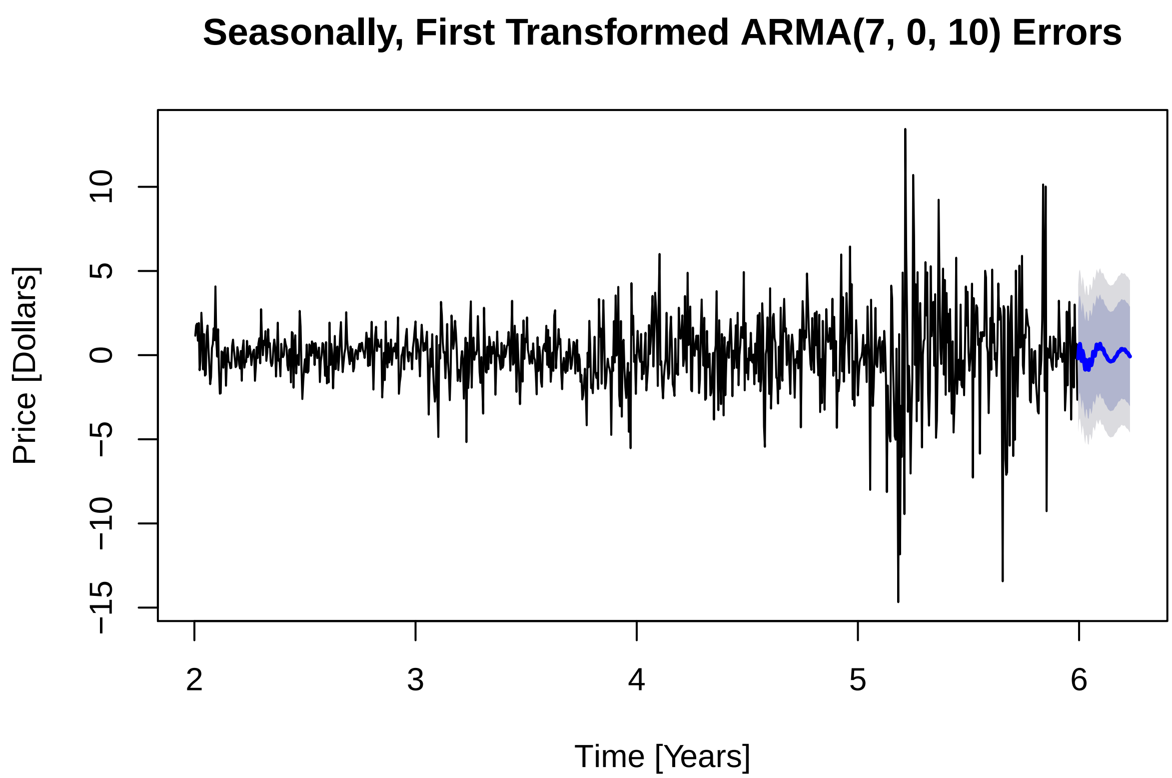
```
order = final.order
modelTot3 = Arima(high_ts$high_tot_ts, order = order, xreg = high_ts$period2)
```

Forecasting

Forecast values according to seasonally and first transformed ARMA model. Take the more extreme of the upper and lower end of the forecast's confidence interval to better simulate the 'fizziness' of the market. An example forecast plot is shown.

```
futureXReg = as.matrix(rep(0, 60))

modelA_forecast = forecast(modelA1, xreg = as.matrix(futureXReg))[c('upper', 'lower')]
modelB_forecast = forecast(modelB1, xreg = as.matrix(futureXReg))[c('upper', 'lower')]
modelC_forecast = forecast(modelC1, xreg = as.matrix(futureXReg))[c('upper', 'lower')]
modelHealth_forecast = forecast(modelHealth1, xreg = as.matrix(futureXReg))[c('upper', 'lower')]
modelConstpl_forecast = forecast(modelConstpl1, xreg = as.matrix(futureXReg))[c('upper', 'lower')]
modelNrg_forecast = forecast(modelNrg1, xreg = as.matrix(futureXReg))[c('upper', 'lower')]

modelA1_pred = c()
modelB1_pred = c()
modelC1_pred = c()
modelHealth1_pred = c()
modelConstpl1_pred = c()
modelNrg1_pred = c()

for (i in 1:length(modelA_forecast$upper[,1])) {
  if (modelA_forecast$upper[i,2] > abs(modelA_forecast$lower[i,2])) {
    modelA1_pred = c(modelA1_pred, modelA_forecast$upper[i,2])
  }
  else {
    modelA1_pred = c(modelA1_pred, modelA_forecast$lower[i,2])
  }

  if (modelB_forecast$upper[i,2] > abs(modelB_forecast$lower[i,2])) {
```

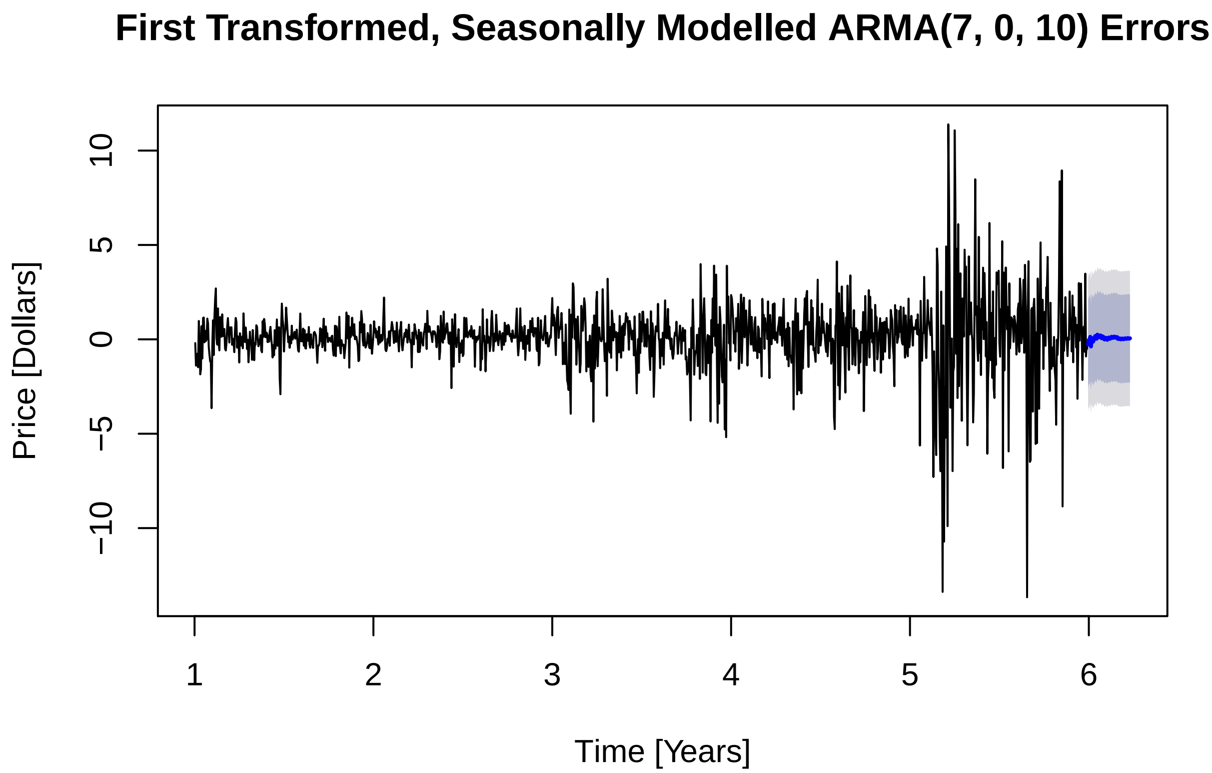
```
    modelB1_pred = c(modelB1_pred, modelB_forecast$upper[i,2])
  }
  else {
    modelB1_pred = c(modelB1_pred, modelB_forecast$lower[i,2])
  }

  if (modelC_forecast$upper[i,2] > abs(modelC_forecast$lower[i,2])) {
    modelC1_pred = c(modelC1_pred, modelC_forecast$upper[i,2])
  }
  else {
    modelC1_pred = c(modelC1_pred, modelC_forecast$lower[i,2])
  }

  if (modelHealth_forecast$upper[i,2] > abs(modelHealth_forecast$lower[i,2])) {
    modelHealth1_pred = c(modelHealth1_pred, modelHealth_forecast$upper[i,2])
  }
  else {
    modelHealth1_pred = c(modelHealth1_pred, modelHealth_forecast$lower[i,2])
  }

  if (modelConstpl_forecast$upper[i,2] > abs(modelConstpl_forecast$lower[i,2])) {
    modelConstpl1_pred = c(modelConstpl1_pred, modelConstpl_forecast$upper[i,2])
  }
  else {
    modelConstpl1_pred = c(modelConstpl1_pred, modelConstpl_forecast$lower[i,2])
  }

  if (modelNrg_forecast$upper[i,2] > abs(modelNrg_forecast$lower[i,2])) {
    modelNrg1_pred = c(modelNrg1_pred, modelNrg_forecast$upper[i,2])
  }
  else {
    modelNrg1_pred = c(modelNrg1_pred, modelNrg_forecast$lower[i,2])
  }
}

plot(forecast(modelA1, xreg = as.matrix(futureXReg)), main='Seasonally, First Transformed ARMA(7, 0, 10)
```



Forecast values according to first-differenced and seasonally modelled ARMA model. Take the more extreme of the upper and lower end of the forecast's confidence interval to better simulate the 'fizziness' of the market. An example forecast plot is shown.

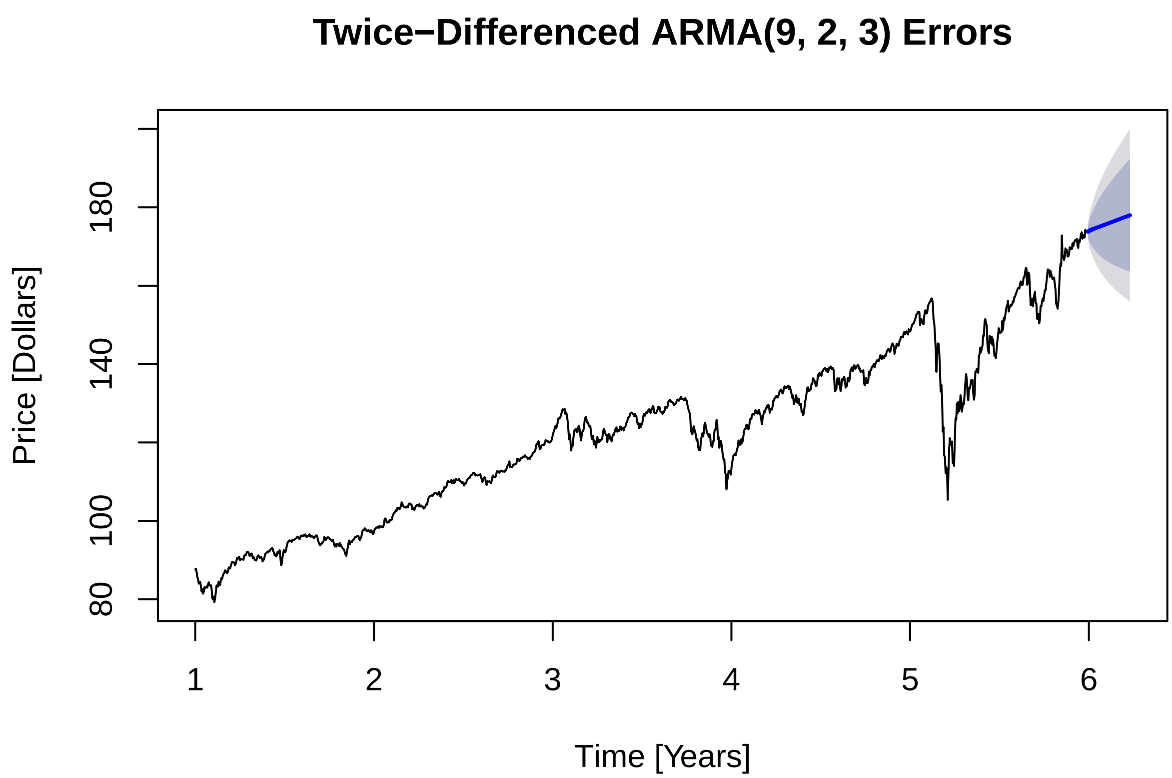
```
futureXReg = as.matrix(cbind(rep(0, 60), c(rep(0, 42), rep(1, 18)), rep(0, 60), rep(0, 60)))

modelA_forecast = forecast(modelA2, xreg = as.matrix(futureXReg))[c('upper', 'lower')]
modelB_forecast = forecast(modelB2, xreg = as.matrix(futureXReg))[c('upper', 'lower')]
modelC_forecast = forecast(modelC2, xreg = as.matrix(futureXReg))[c('upper', 'lower')]
modelHealth_forecast = forecast(modelHealth2, xreg = as.matrix(futureXReg))[c('upper', 'lower')]
modelConstpl_forecast = forecast(modelConstpl2, xreg = as.matrix(futureXReg))[c('upper', 'lower')]
modelNrg_forecast = forecast(modelNrg2, xreg = as.matrix(futureXReg))[c('upper', 'lower')]

modelA2_pred = c()
modelB2_pred = c()
modelC2_pred = c()
modelHealth2_pred = c()
modelConstpl2_pred = c()
modelNrg2_pred = c()

for (i in 1:length(modelA_forecast$upper[,1])) {
  if (modelA_forecast$upper[i,2] > abs(modelA_forecast$lower[i,2])) {
    modelA2_pred = c(modelA2_pred, modelA_forecast$upper[i,2])
  }
  else {
    modelA2_pred = c(modelA2_pred, modelA_forecast$lower[i,2])
  }
```

```
  if (modelB_forecast$upper[i,2] > abs(modelB_forecast$lower[i,2])) {
    modelB2_pred = c(modelB2_pred, modelB_forecast$upper[i,2])
  }
  else {
    modelB2_pred = c(modelB2_pred, modelB_forecast$lower[i,2])
  }

  if (modelC_forecast$upper[i,2] > abs(modelC_forecast$lower[i,2])) {
    modelC2_pred = c(modelC2_pred, modelC_forecast$upper[i,2])
  }
  else {
    modelC2_pred = c(modelC2_pred, modelC_forecast$lower[i,2])
  }

  if (modelHealth_forecast$upper[i,2] > abs(modelHealth_forecast$lower[i,2])) {
    modelHealth2_pred = c(modelHealth2_pred, modelHealth_forecast$upper[i,2])
  }
  else {
    modelHealth2_pred = c(modelHealth2_pred, modelHealth_forecast$lower[i,2])
  }

  if (modelConstpl_forecast$upper[i,2] > abs(modelConstpl_forecast$lower[i,2])) {
    modelConstpl2_pred = c(modelConstpl2_pred, modelConstpl_forecast$upper[i,2])
  }
  else {
    modelConstpl2_pred = c(modelConstpl2_pred, modelConstpl_forecast$lower[i,2])
  }

  if (modelNrg_forecast$upper[i,2] > abs(modelNrg_forecast$lower[i,2])) {
    modelNrg2_pred = c(modelNrg2_pred, modelNrg_forecast$upper[i,2])
  }
  else {
    modelNrg2_pred = c(modelNrg2_pred, modelNrg_forecast$lower[i,2])
  }
}

plot(forecast(modelA2, xreg = as.matrix(futureXReg)), main='First Transformed, Seasonally Modelled ARMA
```



Forecast and plot values according to twice-differenced ARMA model.

```
futureXReg = as.matrix(rep(0, 60))

modelTot3_pred = forecast(modelTot3, xreg = as.matrix(futureXReg))$mean

plot(forecast(modelTot3, xreg = as.matrix(futureXReg)), main='Twice-Differenced ARMA(9, 2, 3) Errors', :
```



Use Ljung-Box test to validate models lagged by twice the supposed seasonal period.

```r
lag = 504

print('Seasonally and First Differenced:')
```

```
## [1] "Seasonally and First Differenced:"
```

```r
Box.test(resid(modelA1), lag = lag, type = 'Ljung-Box')$p.value
```

```
## [1] 0.2386062
```

```r
Box.test(resid(modelB1), lag = lag, type = 'Ljung-Box')$p.value
```

```
## [1] 0.0006602894
```

```r
Box.test(resid(modelC1), lag = lag, type = 'Ljung-Box')$p.value
```

```
## [1] 0.05973469
```

```r
Box.test(resid(modelHealth1), lag = lag, type = 'Ljung-Box')$p.value
```

```
## [1] 0.0004655338
```

```r
Box.test(resid(modelConstpl1), lag = lag, type = 'Ljung-Box')$p.value
```

```
## [1] 0.01796159
```

```r
Box.test(resid(modelNrg1), lag = lag, type = 'Ljung-Box')$p.value
```

```
## [1] 4.726064e-08
```

```r
print('First Differenced and Seasonally Modelled:')
```

```
## [1] "First Differenced and Seasonally Modelled:"
```

```r
Box.test(resid(modelA2), lag = lag, type = 'Ljung-Box')$p.value
```

```
## [1] 0.9867834
```

```r
Box.test(resid(modelB2), lag = lag, type = 'Ljung-Box')$p.value
```

```
## [1] 0.9389261
```

```r
Box.test(resid(modelC2), lag = lag, type = 'Ljung-Box')$p.value
```

```
## [1] 0.9196863
```

```r
Box.test(resid(modelHealth2), lag = lag, type = 'Ljung-Box')$p.value
```

```
## [1] 0.8121747
```

```r
Box.test(resid(modelConstpl2), lag = lag, type = 'Ljung-Box')$p.value
```

```
## [1] 0.9960634
```

```r
Box.test(resid(modelNrg2), lag = lag, type = 'Ljung-Box')$p.value
```

```
## [1] 0.9939127
```

```r
print('Twice-Differenced:')
```

```
## [1] "Twice-Differenced:"
```

```r
Box.test(resid(modelTot3), lag = lag, type = 'Ljung-Box')$p.value
```

```
## [1] 0.9620877
```

Use cumulative sum method to forecast re-adjsted future values from last known training point. A fourth series is created by averaging other three forecasts.

```r
y = high_ts$high_tot_ts
pred = rowMeans(cbind(modelA1_pred, modelB1_pred, modelC1_pred, modelHealth1_pred, modelConstpl1_pred, m
for (t in 1260:1319) {
  trans = pred[t-1259] + y[t-252] - y[t-253] + y[t-1]
  y = c(y, trans)
}
cumsumSeries1 = y[1260:1319]

cumsumSeries2 = cumsum(c(high_ts$high_tot_ts[1259],
                        rowMeans(
                              cbind(
                                modelA2_pred, modelB2_pred, modelC2_pred, modelHealth2_pred, modelConstp
                                ))))[-1]

cumsumSeries3 = modelTot3_pred

cumsumSeries4 = rowMeans(cbind(cumsumSeries1, cumsumSeries2, cumsumSeries3))
```

Calculate MSE for each prediction set.

```r
mean((high_avgs$mean_tot[which(as.Date(high_avgs$date) > '2020-01-01')][1:60] - cumsumSeries1)^2)
```

```
## [1] 368.5428
```

```r
mean((high_avgs$mean_tot[which(as.Date(high_avgs$date) > '2020-01-01')][1:60] - cumsumSeries2)^2)
```

```
## [1] 566.126
```

```r
mean((high_avgs$mean_tot[which(as.Date(high_avgs$date) > '2020-01-01')][1:60] - cumsumSeries3)^2)
```

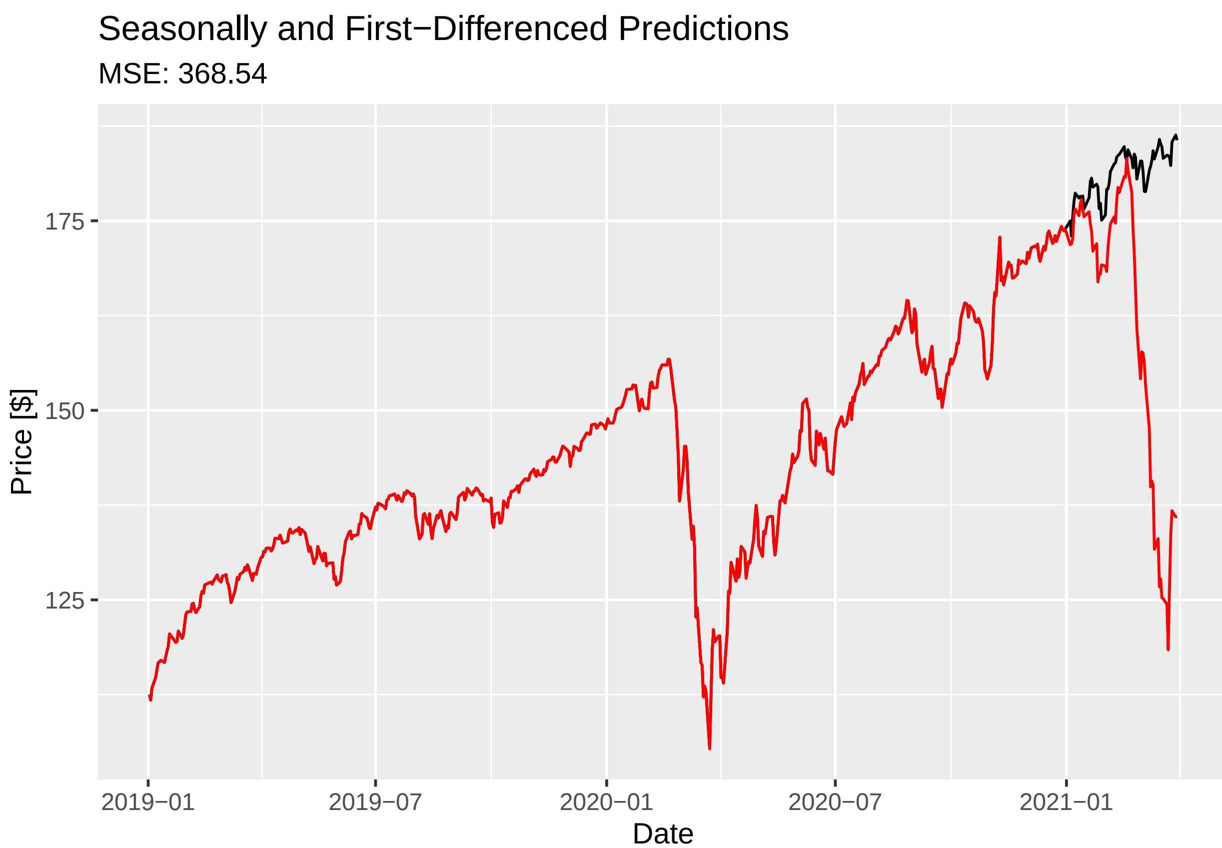
```
## [1] 1334.863
```

```r
mean((high_avgs$mean_tot[which(as.Date(high_avgs$date) > '2020-01-01')][1:60] - cumsumSeries4)^2)
```

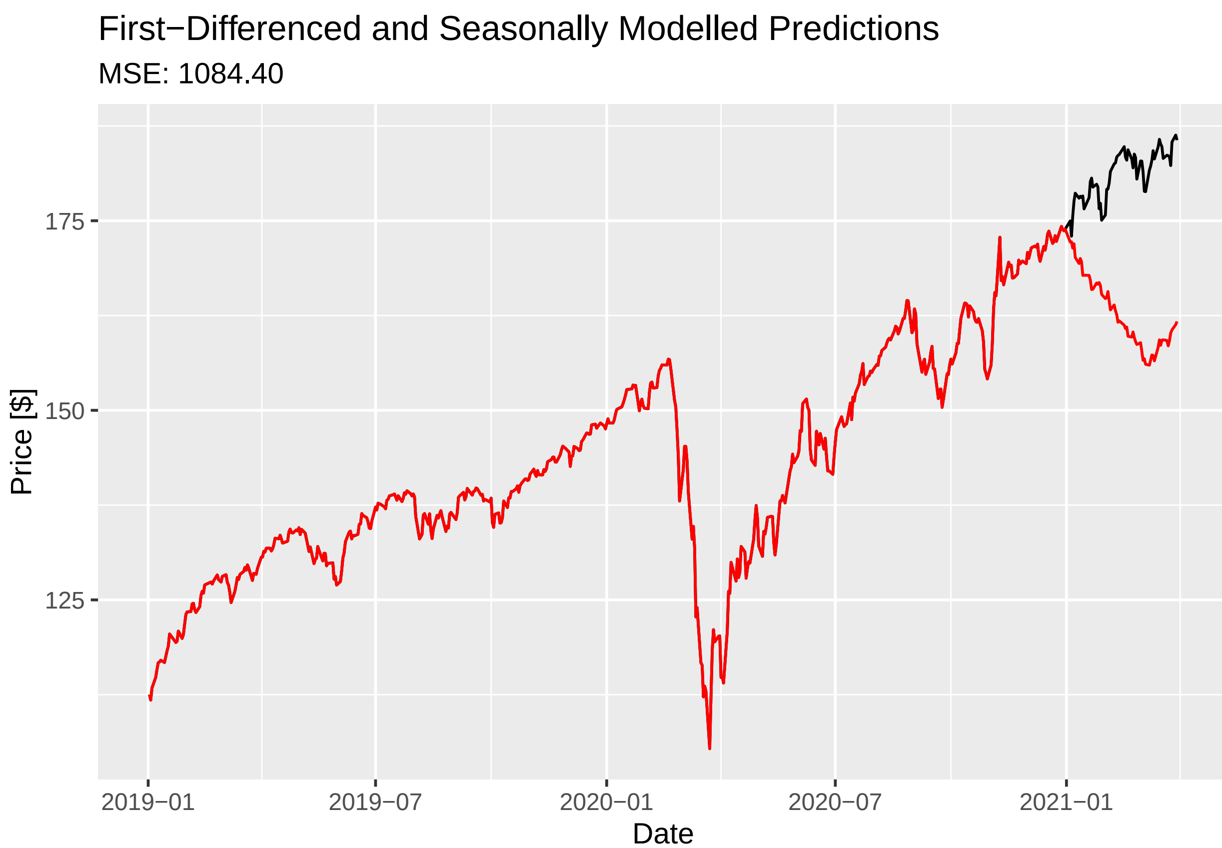
```
## [1] 629.3523
```

Plot aggregated time series and mean_tot time series that it simulates.

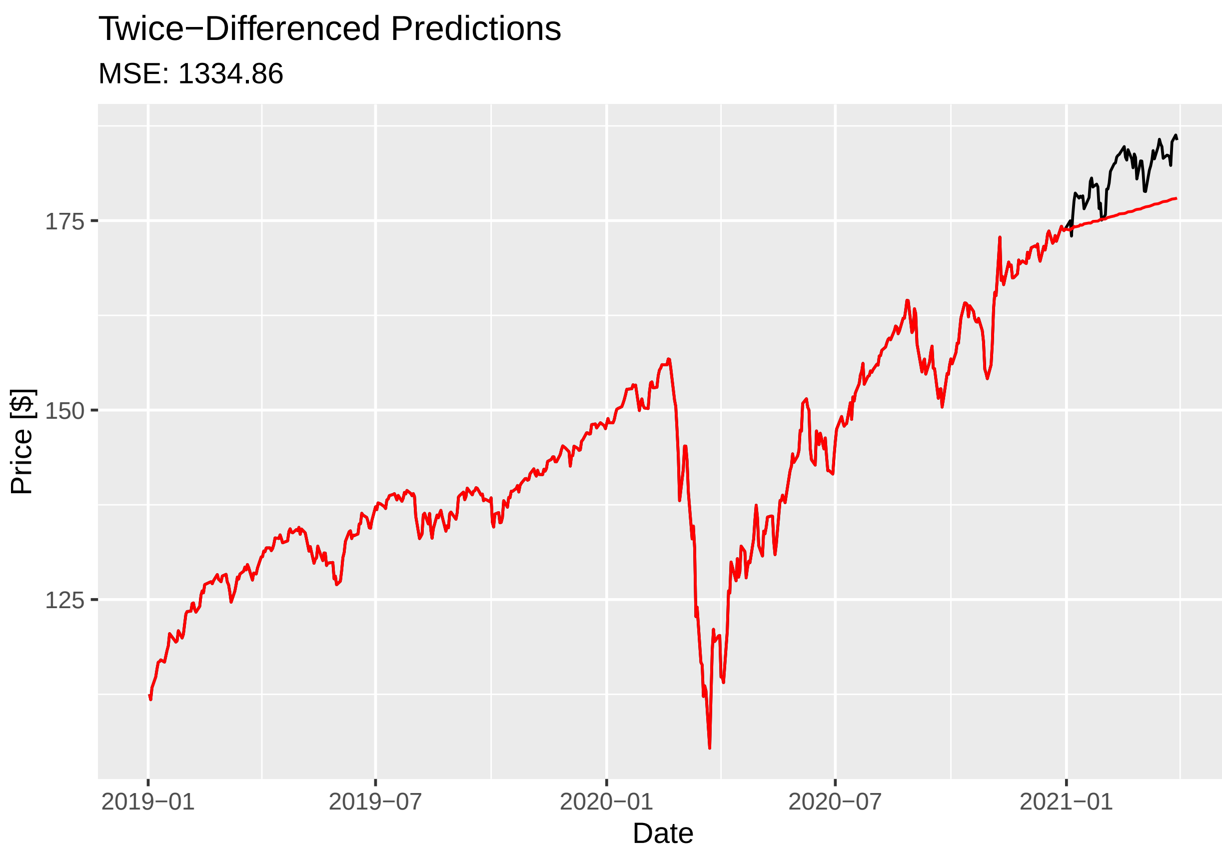
```r
ggplot(mapping = aes(x = as.Date(high_avgs$date)[which(as.Date(high_avgs$date) > '2019-01-01')])) +
  geom_path(aes(y = high_avgs$mean_tot[which(as.Date(high_avgs$date) > '2019-01-01')])) +
  geom_path(aes(y = c(high_ts$high_tot_ts[which(as.Date(high_ts$date) > '2019-01-01')], cumsumSeries1))
  labs(title = 'Seasonally and First-Differenced Predictions', subtitle = 'MSE: 368.54', x = 'Date', y =
```



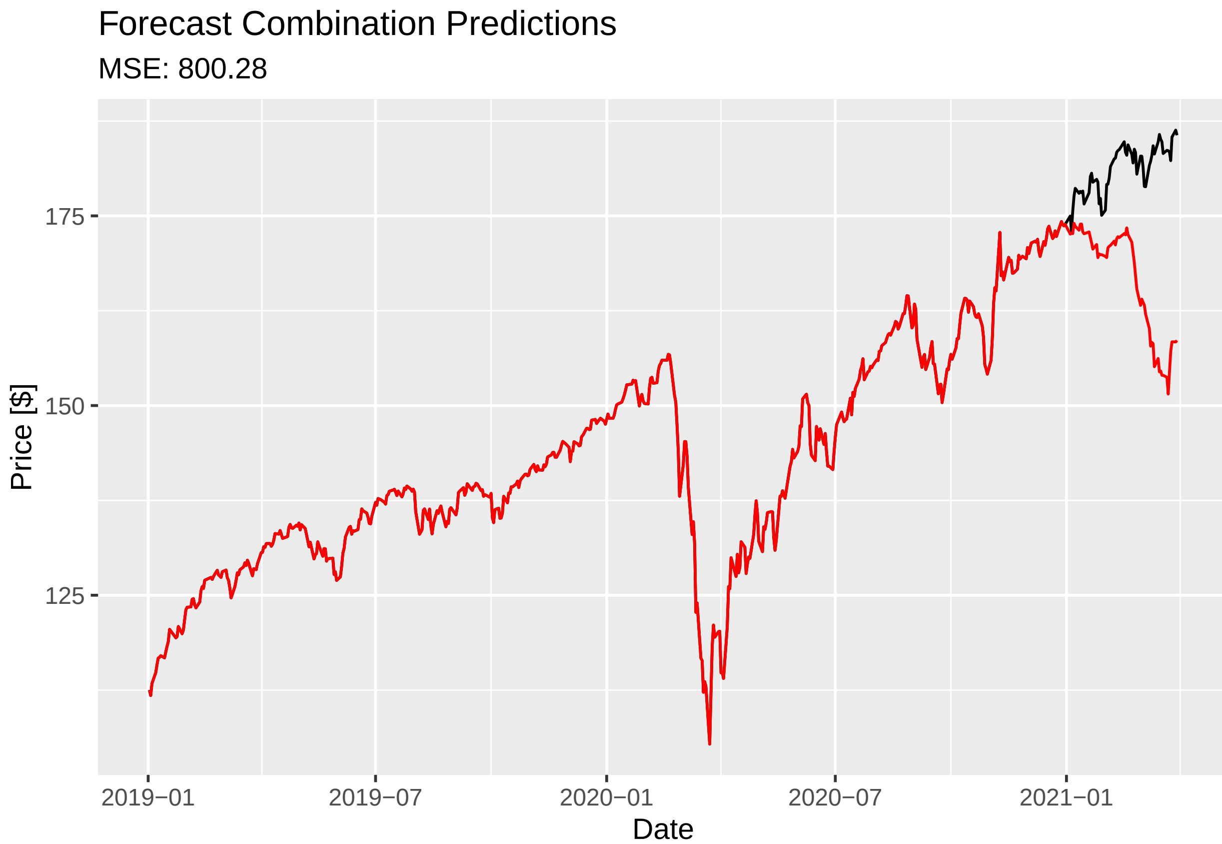

```
ggplot(mapping = aes(x = as.Date(high_avgs$date)[which(as.Date(high_avgs$date) > '2019-01-01')])) +
  geom_path(aes(y = high_avgs$mean_tot[which(as.Date(high_avgs$date) > '2019-01-01')])) +
  geom_path(aes(y = c(high_ts$high_tot_ts[which(as.Date(high_ts$date) > '2019-01-01')], cumsumSeries2))
  labs(title = 'First-Differenced and Seasonally Modelled Predictions', subtitle = 'MSE: 1084.40', x =
```




```
ggplot(mapping = aes(x = as.Date(high_avgs$date)[which(as.Date(high_avgs$date) > '2019-01-01')])) +
  geom_path(aes(y = high_avgs$mean_tot[which(as.Date(high_avgs$date) > '2019-01-01')])) +
  geom_path(aes(y = c(high_ts$high_tot_ts[which(as.Date(high_ts$date) > '2019-01-01')], cumsumSeries3))
  labs(title = 'Twice-Differenced Predictions', subtitle = 'MSE: 1334.86', x = 'Date', y = 'Price [$]')
```




```
ggplot(mapping = aes(x = as.Date(high_avgs$date)[which(as.Date(high_avgs$date) > '2019-01-01')])) +
  geom_path(aes(y = high_avgs$mean_tot[which(as.Date(high_avgs$date) > '2019-01-01')])) +
  geom_path(aes(y = c(high_ts$high_tot_ts[which(as.Date(high_ts$date) > '2019-01-01')], cumsumSeries4))
  labs(title = 'Forecast Combination Predictions', subtitle = 'MSE: 800.28', x = 'Date', y = 'Price [$]
```



Forecast Combination Predictions
MSE: 800.28
175
150
125
Price [$]
2019−01
2019−07
2020−01
2020−07
2021−01
Date