analysis_base_first_date

May 15, 2021

1 Analysis of stock prices in different time periods

NOTE: `base date point` means that base value will be set to the first date in dataset.

Example: if we want to get daily prices within a week then `base date point` means that the base value will be set **only** for data point with first date

[1]:

```
import sys

sys.path.append('..')

from analysis_base_first_date import Column
from common import plot, YahooRange

from loguru import logger
import numpy as np
import pandas as pd
from seaborn import lineplot, barplot, scatterplot, boxplot
from matplotlib import pyplot

pd.options.mode.chained_assignment = None

FILENAME = "sp500/sp500.csv"
LIMIT = None

logger.remove()
logger.add(sys.stdout, level="INFO")

pass
```

1.1 Monthly stock price fluctuations within a year

[2]:

```
from analysis_base_first_date import get_best_month

df = get_best_month(FILENAME, YahooRange.YEARS_10, limit=LIMIT)
df
```

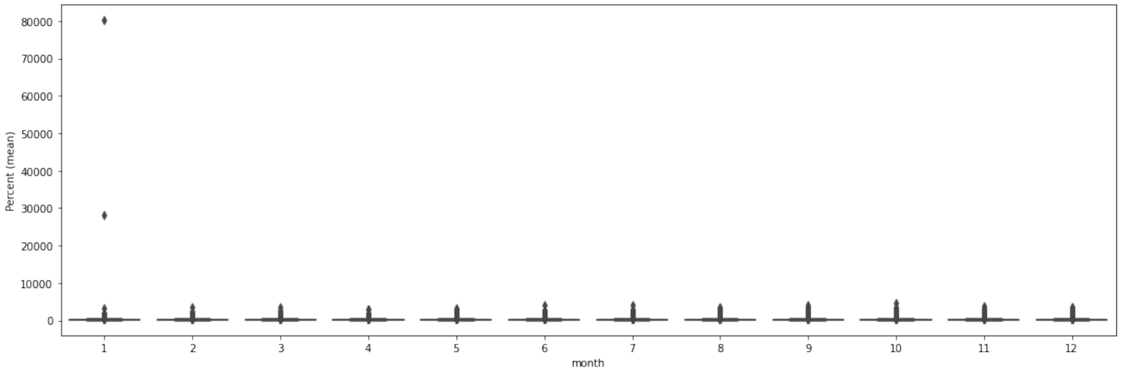
[2]:

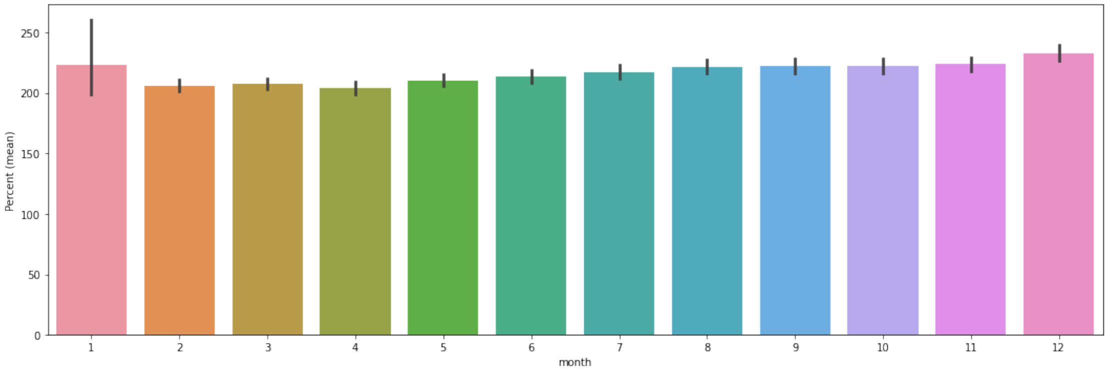
```
       year  month Symbol  Percent (mean)
0      2011      1    MHK           100.0
1      2011      2    MHK       97.123051
2      2011      3    MHK      101.039859
3      2011      4    MHK      107.036391
4      2011      5    MHK      104.592717
…       …     …     …              …
57757  2020      8   EXPD      153.140878
57758  2020      9   EXPD      161.111102
57759  2020     10   EXPD      165.432098
57760  2020     11   EXPD      162.835874
57761  2020     12   EXPD      162.182278

[57762 rows x 4 columns]
```

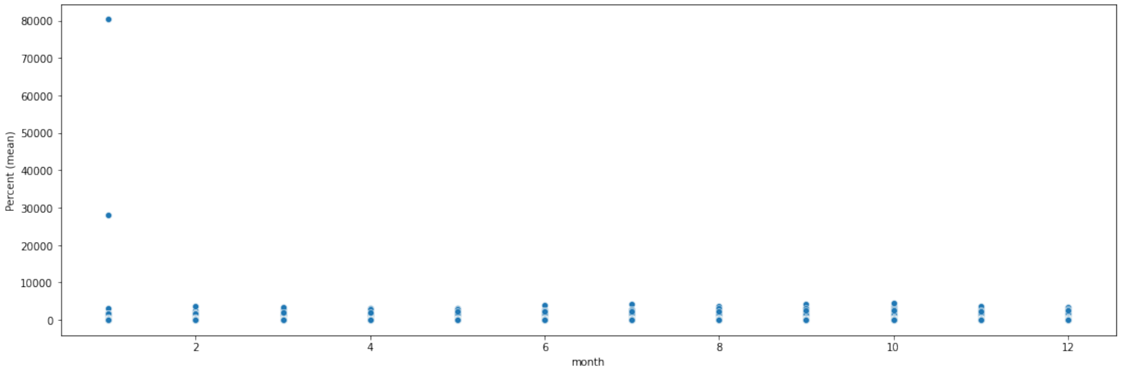
[3]:

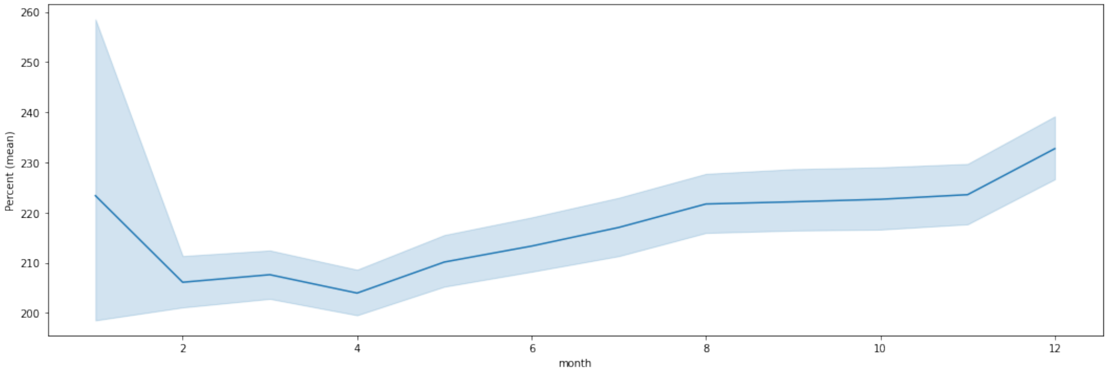
```
plot(x=Column.MONTH, y=Column.PERCENT, data=df)
```

```
       Percent (mean)
month
1          223.354403
2          206.098218
3          207.601802
4          203.939867
5          210.127517
```


80000
70000
60000
50000
40000
30000
20000
10000
0
Percent (mean)
1
2
3
4
5
6
7
8
9
10
11
12
month


250
200
150
100
50
0
Percent (mean)
1
2
3
4
5
6
7
8
9
10
11
12
month


80000
70000
60000
50000
40000
30000
20000
10000
0
Percent (mean)
2
4
6
8
10
12
month


260
250
240
230
220
210
200
Percent (mean)
2
4
6
8
10
12
month

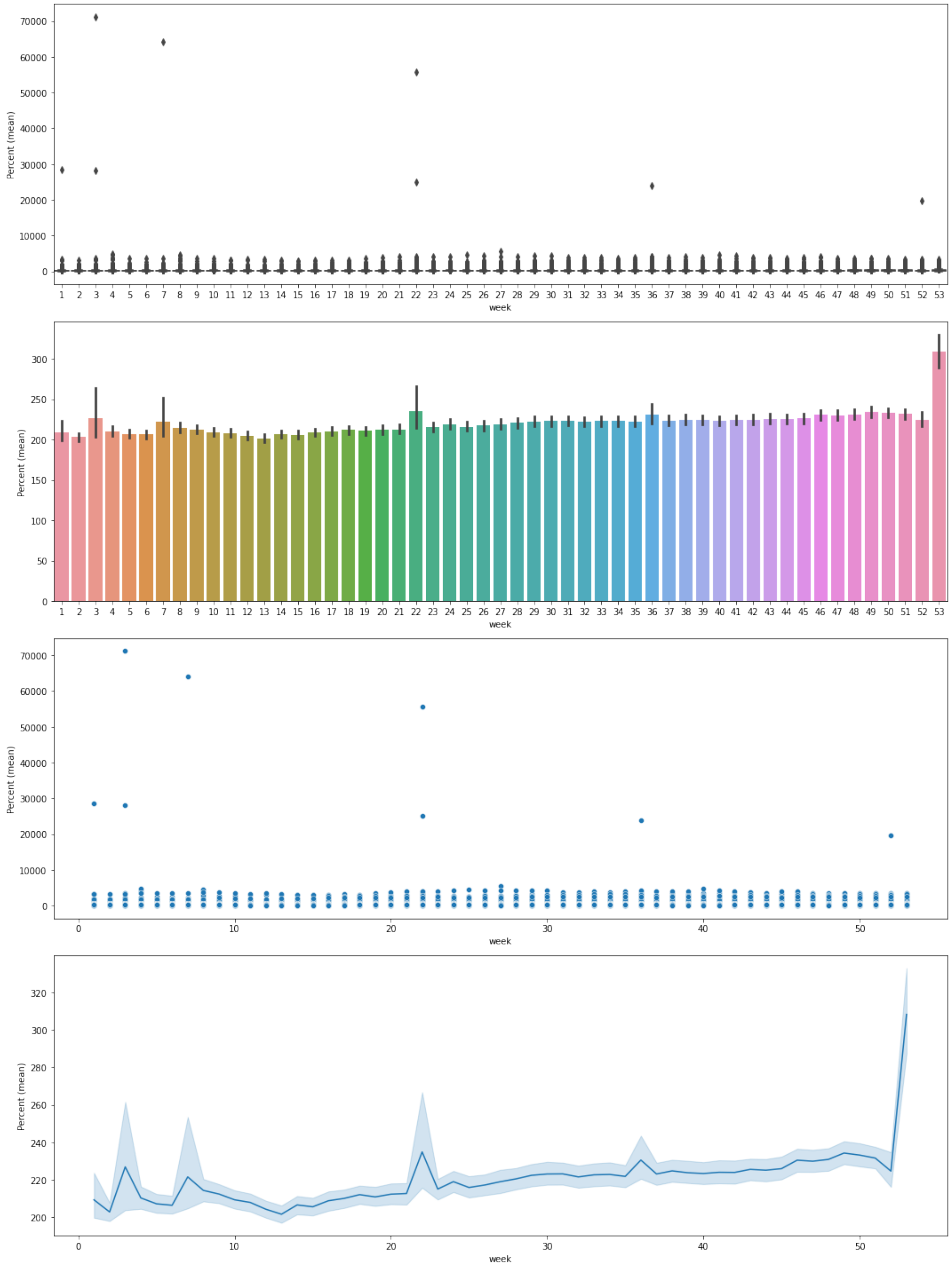
1.2 Weekly stock price fluctuations within a year

```
[4]: from analysis_base_first_date import get_best_week

     df = get_best_week(FILENAME, YahooRange.YEARS_10, limit=LIMIT)

     df
```

```
[4]:         year  week Symbol  Percent (mean)
     0       2014    16   PAYC           100.0
     1       2014    17   PAYC       92.681567
     2       2014    18   PAYC       86.201121
     3       2014    19   PAYC       87.653631
     4       2014    20   PAYC       82.290502
     …        …    …     …              …
     251188  2020    49    NRG      167.573116
     251189  2020    50    NRG      177.116474
     251190  2020    51    NRG      172.498714
     251191  2020    52    NRG      170.856848
     251192  2020    53    NRG      181.682916

     [251193 rows x 4 columns]
```

```
[5]: plot(x=Column.WEEK, y=Column.PERCENT, data=df)
```

```
      Percent (mean)
week
1         209.317289
2          202.84061
3         226.910719
4         210.328463
5         207.183201
```


Percent (mean)
70000
60000
50000
40000
30000
20000
10000
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
week
Percent (mean)
300
250
200
150
100
50
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
week
Percent (mean)
70000
60000
50000
40000
30000
20000
10000
0
0 10 20 30 40 50
week
Percent (mean)
320
300
280
260
240
220
200
0 10 20 30 40 50
week

1.3 Daily stock price fluctuations within a month

[6]:

```
from analysis_base_first_date import get_best_month_day

df = get_best_month_day(FILENAME, YahooRange.YEARS_10, limit=LIMIT)

df
```

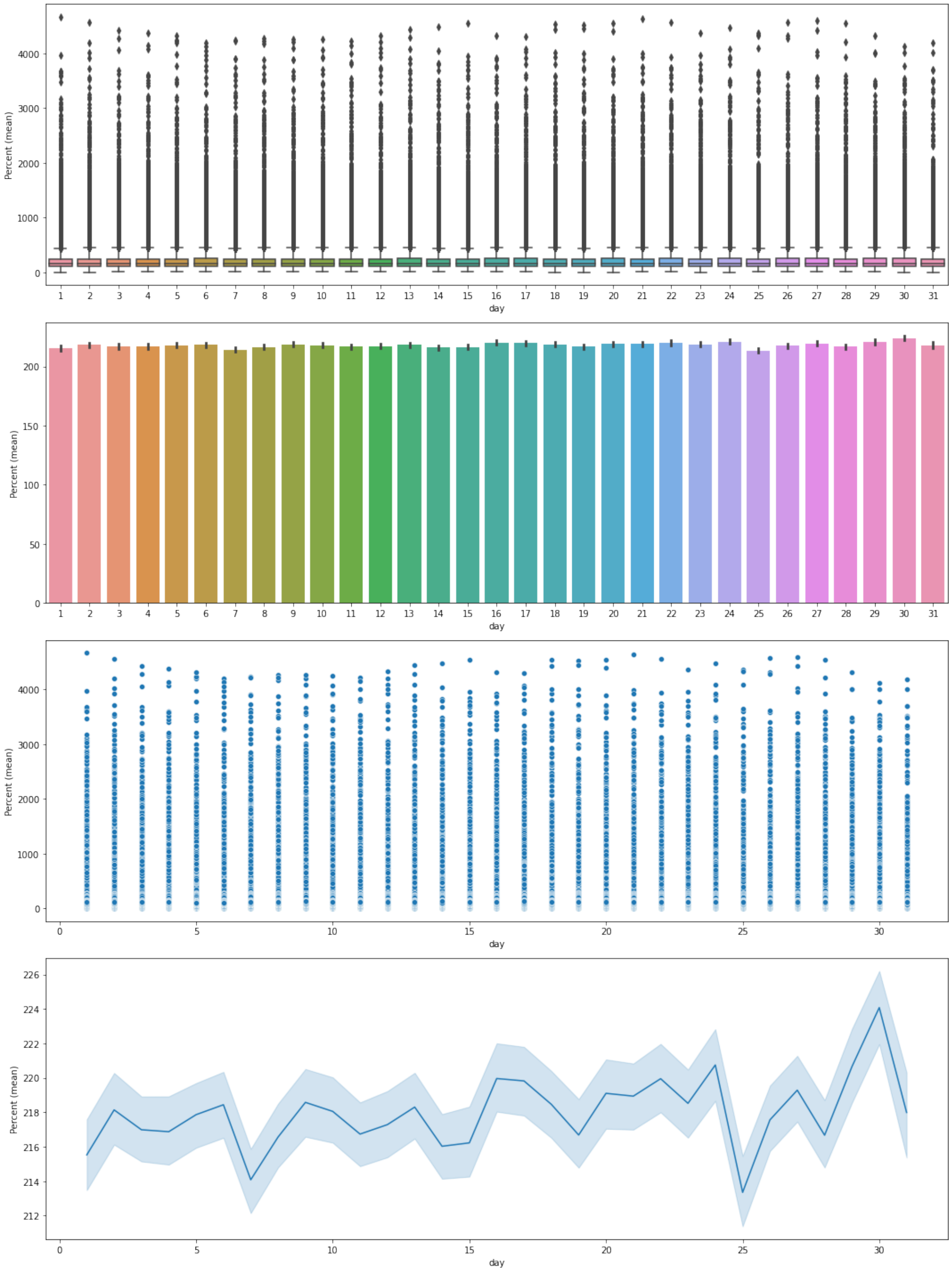
[6]:

```
         year  month  day Symbol  Percent (mean)
0        2010     12   31     EA           100.0
1        2011      1    3     EA       99.457504
2        2011      1    4     EA        98.79445
3        2011      1    5     EA       98.312232
4        2011      1    6     EA       98.251954
…           …     …    …      …               …
1210794  2020     12   24    NUE      119.476409
1210795  2020     12   28    NUE      118.709088
1210796  2020     12   29    NUE      117.828927
1210797  2020     12   30    NUE      116.926201
1210798  2020     12   31    NUE      119.498982

[1210799 rows x 5 columns]
```

[7]:

```
plot(x=Column.DAY, y=Column.PERCENT, data=df)
```

```
     Percent (mean)
day
1         215.527958
2         218.140038
3         216.986208
4         216.873947
5         217.863106
```


Percent (mean)
0
1000
2000
3000
4000
day
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Percent (mean)
0
50
100
150
200
day
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Percent (mean)
0
1000
2000
3000
4000
day
0 5 10 15 20 25 30
Percent (mean)
212
214
216
218
220
222
224
226
day
0 5 10 15 20 25 30

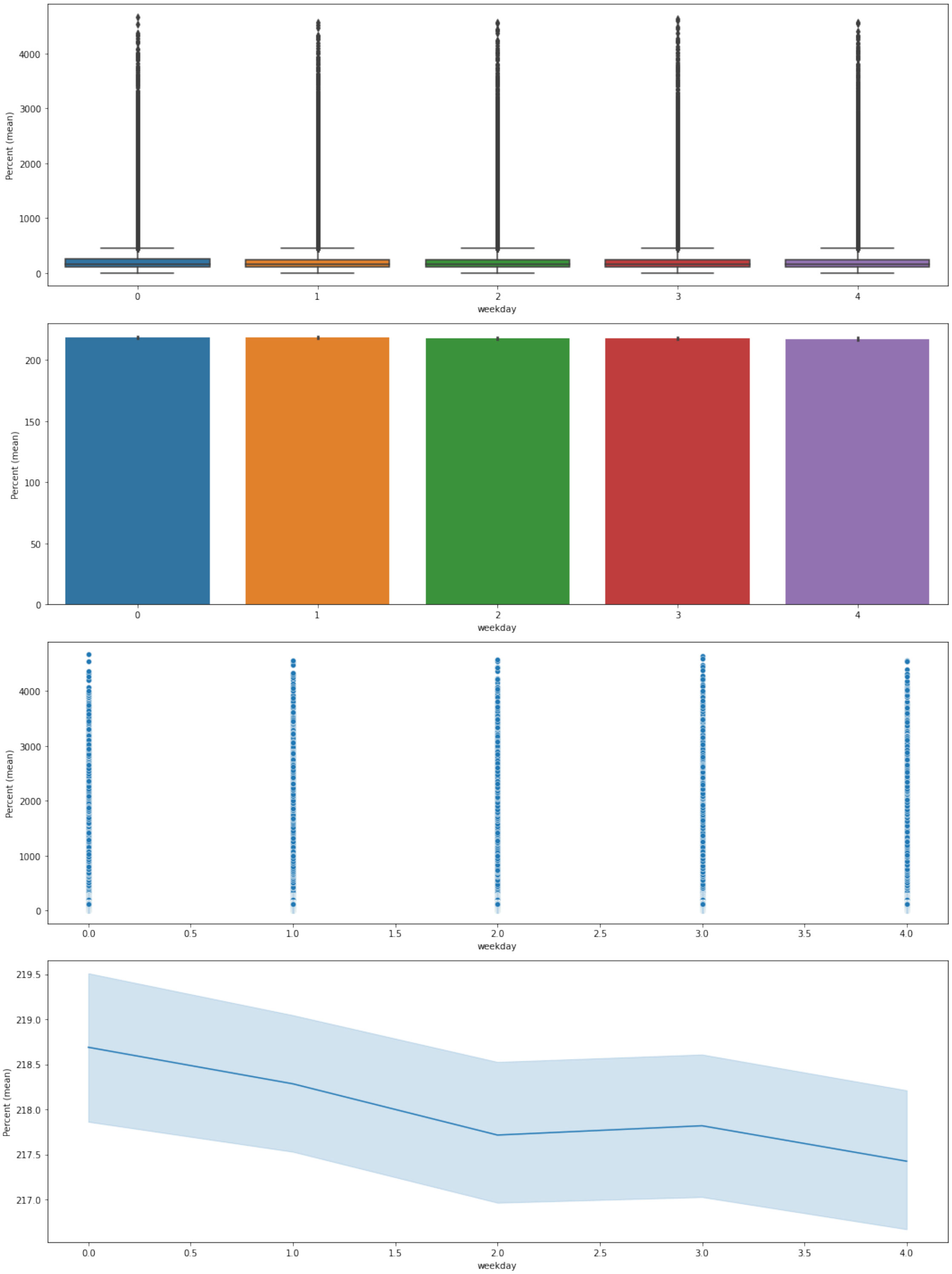
1.4 Daily stock price fluctuations within a week

[8]:
```
from analysis_base_first_date import get_best_weekday


df = get_best_weekday(FILENAME, YahooRange.YEARS_10, limit=LIMIT)

df
```

[8]:
```
         year  week  weekday  Percent (mean)
0        2010    52        4           100.0
1        2011     1        0       99.457504
2        2011     1        1        98.79445
3        2011     1        2       98.312232
4        2011     1        3       98.251954
…           …    …       …              …
1210794  2020    52        3      119.476409
1210795  2020    53        0      118.709088
1210796  2020    53        1      117.828927
1210797  2020    53        2      116.926201
1210798  2020    53        3      119.498982

[1210799 rows x 4 columns]
```

[9]:
```
plot(x=Column.WEEKDAY, y=Column.PERCENT, data=df)
```

```
         Percent (mean)
weekday
0            218.689408
1            218.283829
2            217.715369
3            217.818413
4            217.424899
```


Percent (mean)
4000
3000
2000
1000
0
0
1
2
3
4
weekday
Percent (mean)
200
150
100
50
0
0
1
2
3
4
weekday
Percent (mean)
4000
3000
2000
1000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
weekday
Percent (mean)
219.5
219.0
218.5
218.0
217.5
217.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
weekday

1.5 Hourly stock price fluctuations within a day

[10]:

```
from analysis_base_first_date import get_best_hour

df = get_best_hour(FILENAME, YahooRange.YEARS_2, limit=LIMIT)

df
```

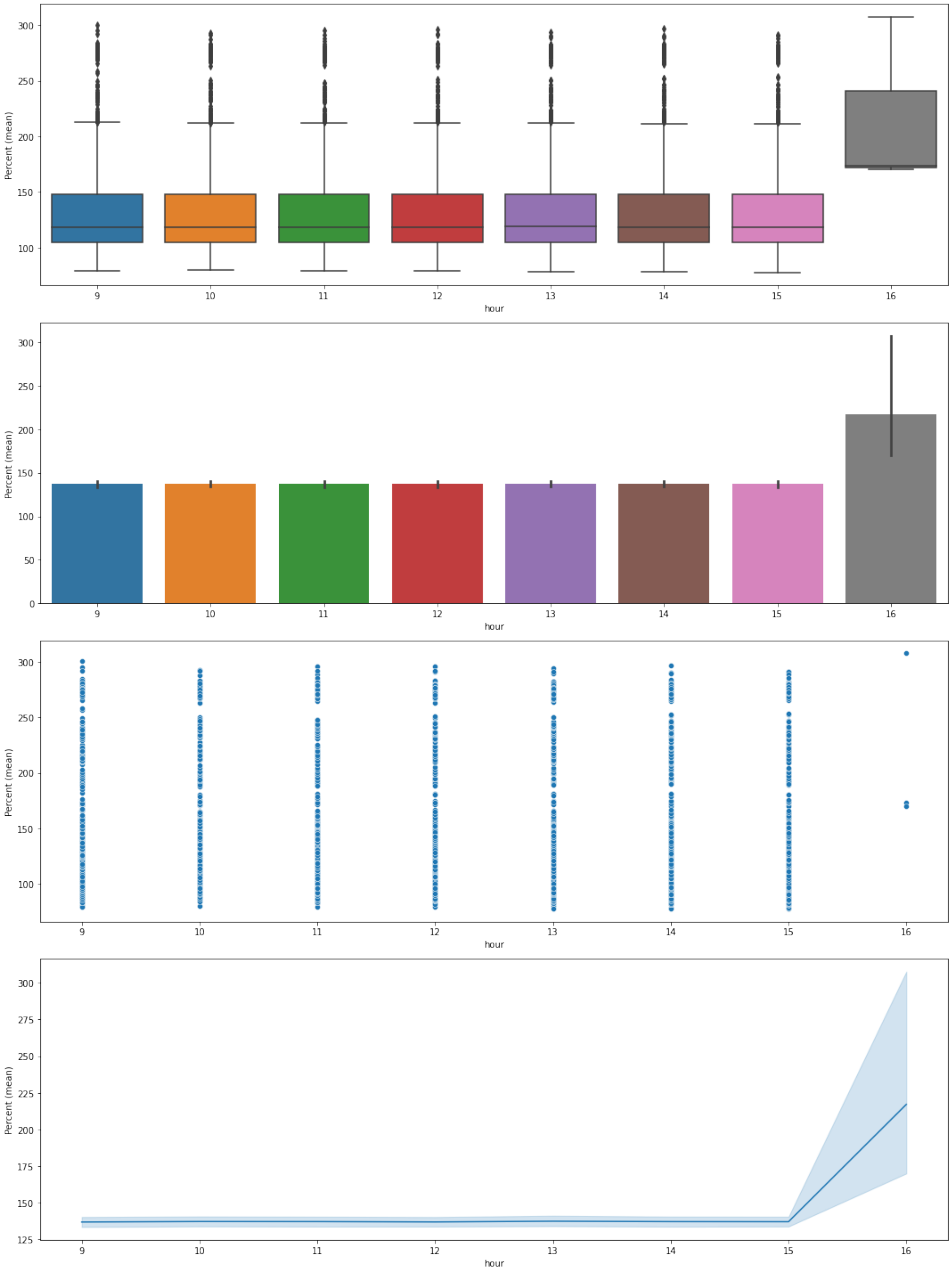
[10]:

```
      year  week  day  hour Symbol  Percent (mean)
0     2020    14    3     9   OTIS           100.0
1     2020    14    3    10   OTIS       99.065634
2     2020    14    3    11   OTIS      104.238834
3     2020    14    3    12   OTIS      106.494983
4     2020    14    3    13   OTIS      111.109845
…      …     …   …    …     …               …
5411  2020    53   31    12   CARR      269.927533
5412  2020    53   31    13   CARR       270.79709
5413  2020    53   31    14   CARR       270.79709
5414  2020    53   31    15   CARR      272.934785
5415  2021    19   12    16   CARR      307.608697

[5416 rows x 6 columns]
```

[11]:

```
plot(x=Column.HOUR, y=Column.PERCENT, data=df)
```

```
      Percent (mean)
hour
9         136.924169
10        137.259433
11        137.208177
12        136.972414
13        137.473135
```


Percent (mean)
300
250
200
150
100
9
10
11
12
13
14
15
16
hour
Percent (mean)
300
250
200
150
100
50
0
9
10
11
12
13
14
15
16
hour
Percent (mean)
300
250
200
150
100
9
10
11
12
13
14
15
16
hour
Percent (mean)
300
275
250
225
200
175
150
125
9
10
11
12
13
14
15
16
hour

1.6 Hourly and quarterly stock price fluctuations within an day

[12]:

```
from analysis_base_first_date import get_best_time

df = get_best_time(FILENAME, YahooRange.DAYS_58, limit=LIMIT)

df
```

```
[*********************100%***********************]  505 of 505 completed

8 Failed downloads:
- NBL: No data found, symbol may be delisted
- BRK.B: No data found, symbol may be delisted
- BF.B: No data found for this date range, symbol may be delisted
- CTL: No data found, symbol may be delisted
- TIF: No data found, symbol may be delisted
- CXO: No data found, symbol may be delisted
- ETFC: No data found, symbol may be delisted
- MYL: No data found, symbol may be delisted
```

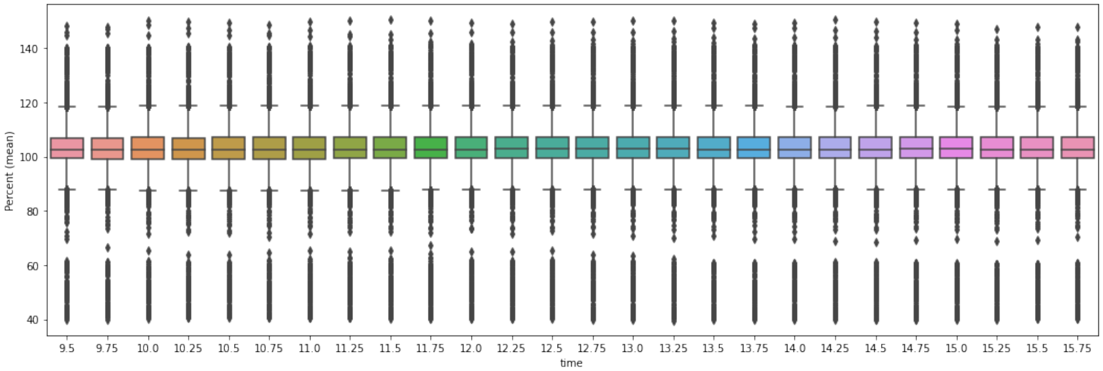
[12]:

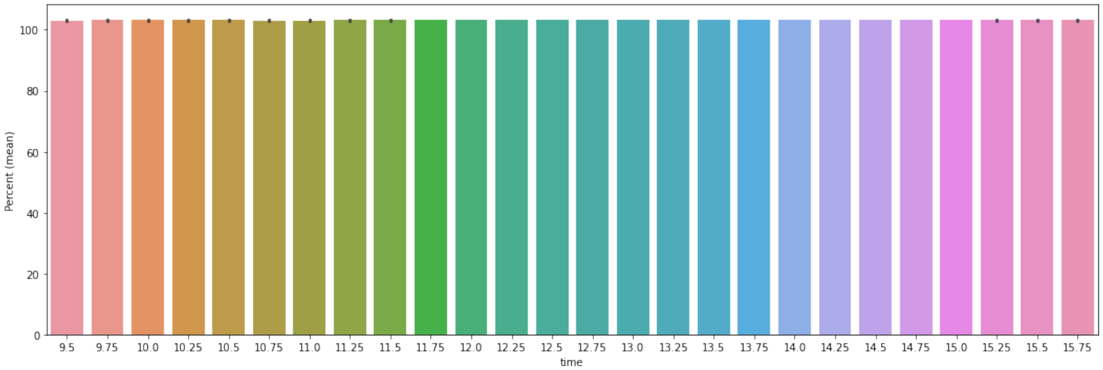
```
        year  week  day  hour  minute   time Symbol  Percent (mean)
0       2021    11   17     9      30    9.5    GPN           100.0
1       2021    11   17     9      45   9.75    GPN      100.050962
2       2021    11   17    10       0   10.0    GPN       99.902694
3       2021    11   17    10      15  10.25    GPN       99.888794
4       2021    11   17    10      30   10.5    GPN       99.990731
…        …    …    …    …       …     …    …              …
511660  2021    19   12    14      45  14.75    RHI      115.898912
511661  2021    19   12    15       0   15.0    RHI      116.063429
511662  2021    19   12    15      15  15.25    RHI      115.727818
511663  2021    19   12    15      30   15.5    RHI      115.254009
511664  2021    19   12    15      45  15.75    RHI      115.464595

[511665 rows x 8 columns]
```

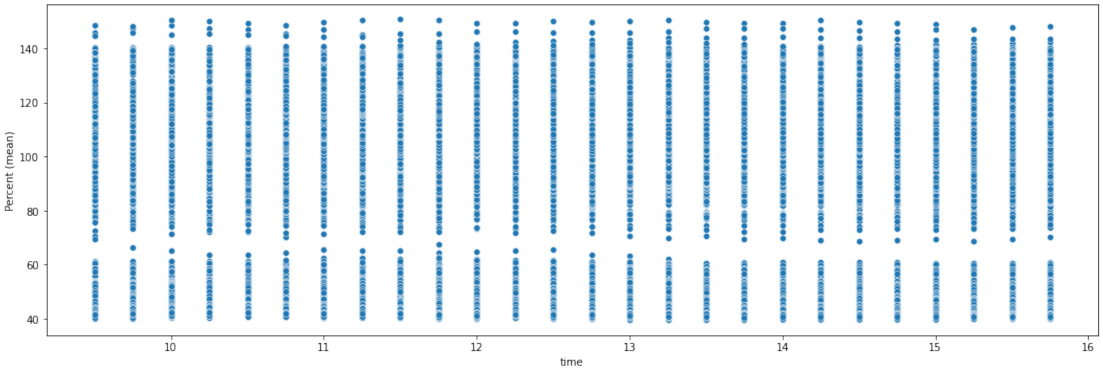
[13]:

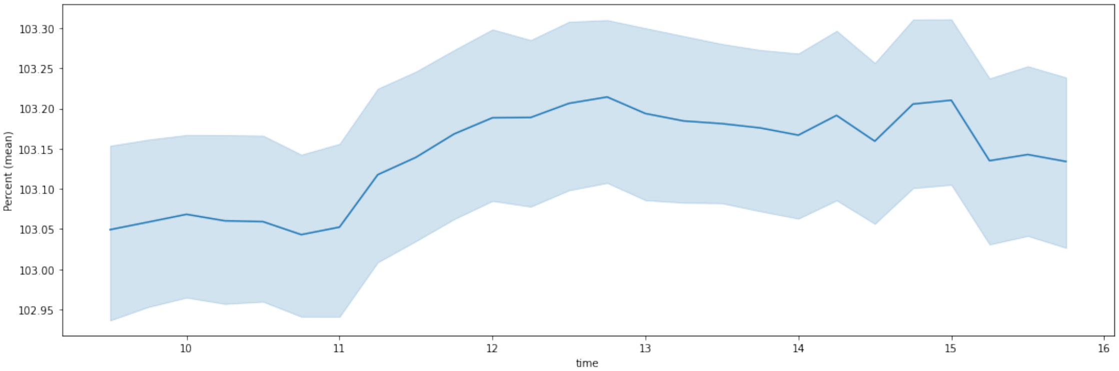
```
plot(x=Column.TIME, y=Column.PERCENT, data=df)
```

```
       Percent (mean)
time
9.50       103.049293
9.75       103.058744
10.00      103.068337
10.25      103.060294
10.50      103.059227
```


Percent (mean)
140
120
100
80
60
40
9.5
9.75
10.0
10.25
10.5
10.75
11.0
11.25
11.5
11.75
12.0
12.25
12.5
12.75
13.0
13.25
13.5
13.75
14.0
14.25
14.5
14.75
15.0
15.25
15.5
15.75
time


Percent (mean)
100
80
60
40
20
0
9.5
9.75
10.0
10.25
10.5
10.75
11.0
11.25
11.5
11.75
12.0
12.25
12.5
12.75
13.0
13.25
13.5
13.75
14.0
14.25
14.5
14.75
15.0
15.25
15.5
15.75
time


Percent (mean)
140
120
100
80
60
40
10
11
12
13
14
15
16
time


Percent (mean)
103.30
103.25
103.20
103.15
103.10
103.05
103.00
102.95
10
11
12
13
14
15
16
time

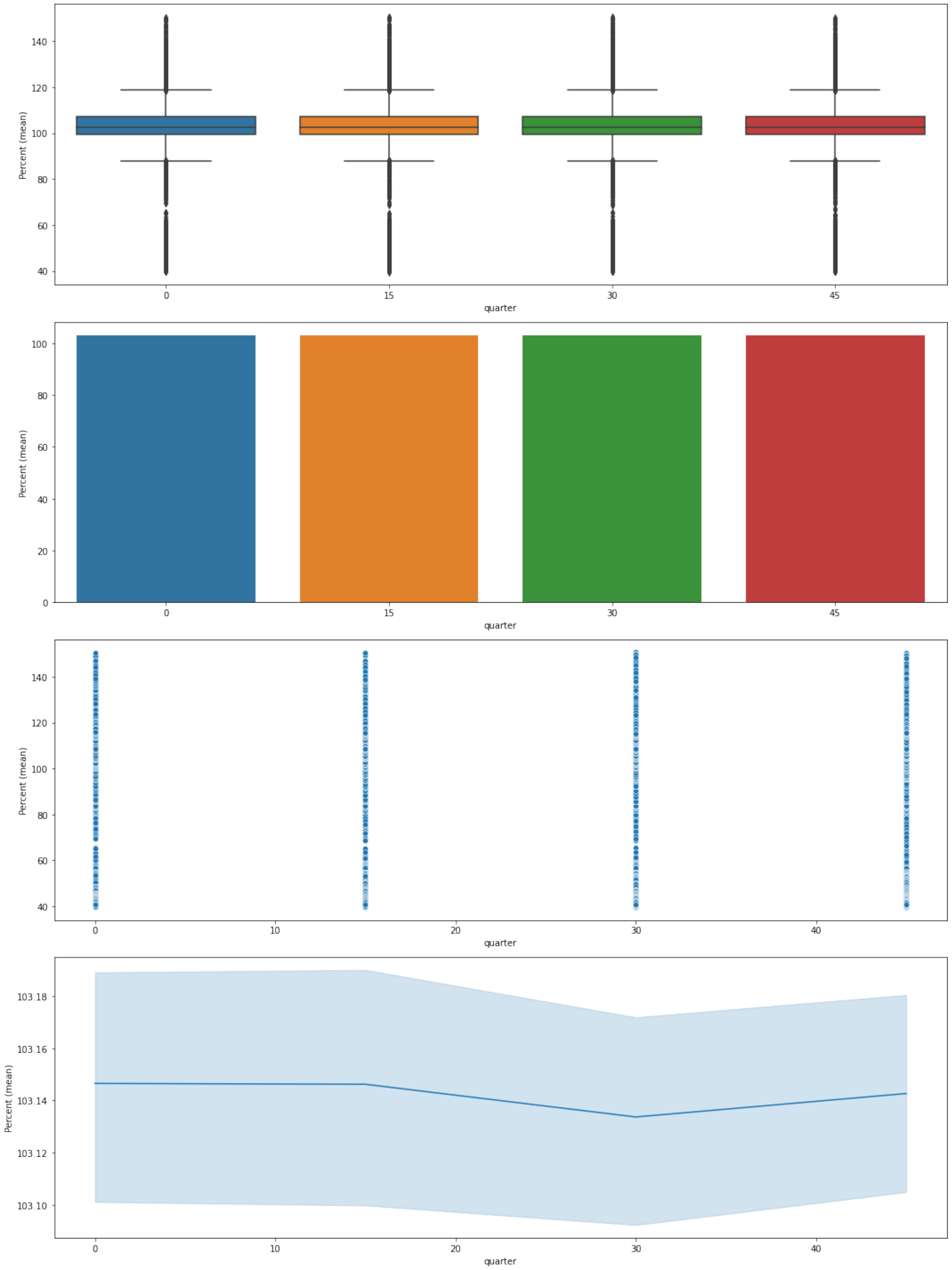
1.7 Quarterly stock price fluctuations within an hour

[14]:
```
from analysis_base_first_date import get_best_quarter

df = get_best_quarter(FILENAME, YahooRange.DAYS_58, limit=LIMIT)

df
```

[14]:
```
        year  week  day  hour  minute  quarter Symbol  Percent (mean)
0       2021    11   17     9      30       30    GPN           100.0
1       2021    11   17     9      45       45    GPN      100.050962
2       2021    11   17    10       0        0    GPN       99.902694
3       2021    11   17    10      15       15    GPN       99.888794
4       2021    11   17    10      30       30    GPN       99.990731
…      …   …  …   …     …      …    …             …
511660  2021    19   12    14      45       45    RHI      115.898912
511661  2021    19   12    15       0        0    RHI      116.063429
511662  2021    19   12    15      15       15    RHI      115.727818
511663  2021    19   12    15      30       30    RHI      115.254009
511664  2021    19   12    15      45       45    RHI      115.464595

[511665 rows x 8 columns]
```

[15]:
```
plot(x=Column.QUARTER, y=Column.PERCENT, data=df)
```

```
         Percent (mean)
quarter
0             103.146544
15            103.146221
30            103.133696
45            103.142677
```


Percent (mean)
140
120
100
80
60
40
0
15
30
45
quarter
Percent (mean)
100
80
60
40
20
0
0
15
30
45
quarter
Percent (mean)
140
120
100
80
60
40
0
10
20
30
40
quarter
Percent (mean)
103.18
103.16
103.14
103.12
103.10
0
10
20
30
40
quarter

[ ]: