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


FILENAME = "dax/dax_mdax_sdax.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
```

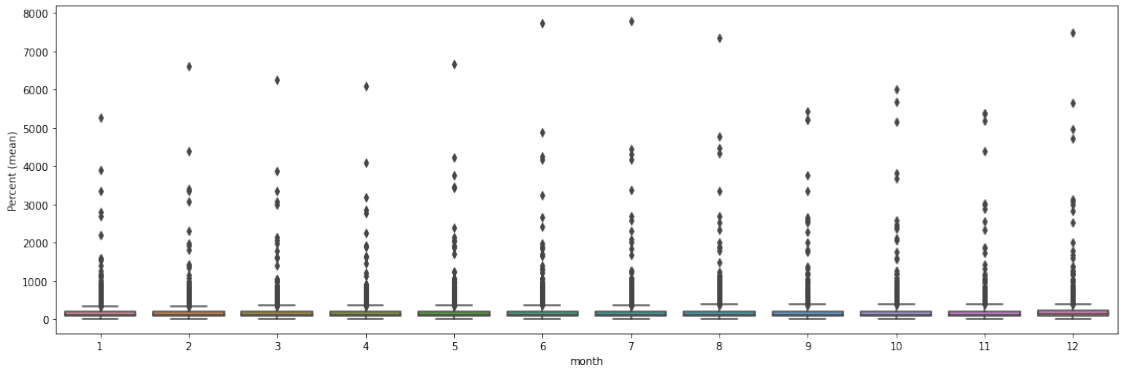
[2]:

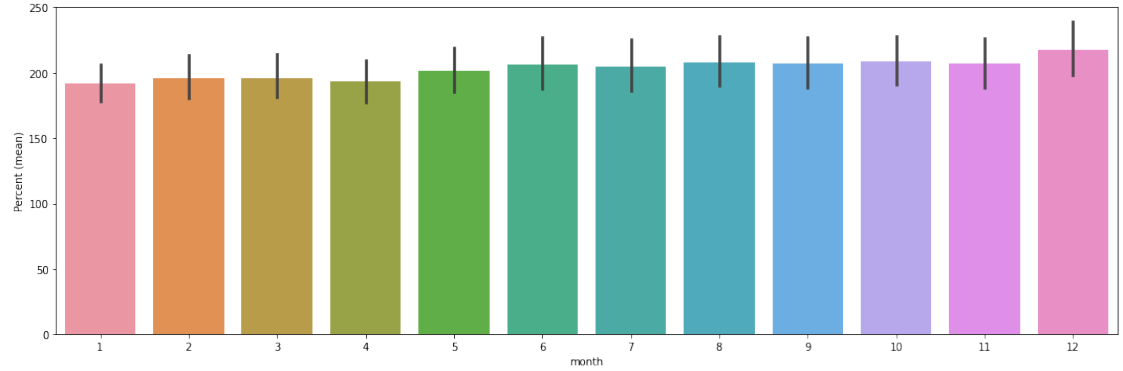
	year	month	Symbol	Percent (mean)
0	2011	1	BVB.DE	100.0

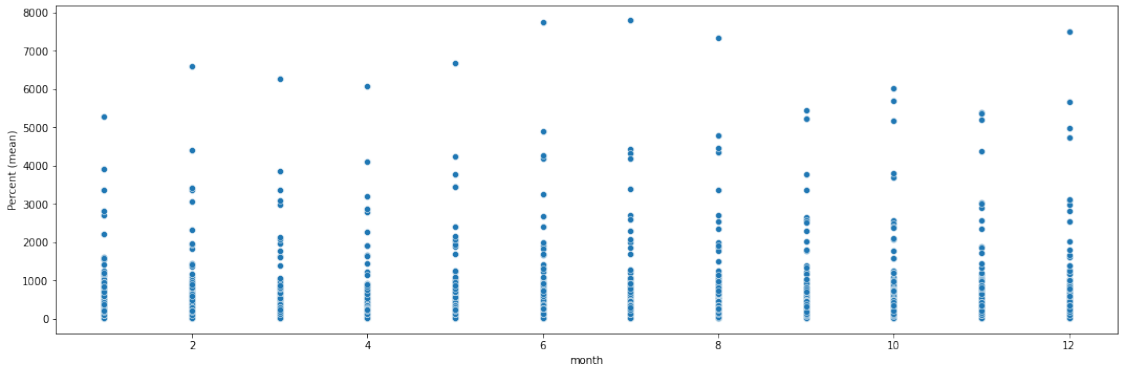
```
1       2011      2  BVB.DE       114.45313
2       2011      3  BVB.DE      117.578127
3       2011      4  BVB.DE      107.812502
4       2011      5  BVB.DE      121.054689
…       …     …      …               …
16042   2020      8  PAT.DE      975.492623
16043   2020      9  PAT.DE      1045.02016
16044   2020     10  PAT.DE      990.240908
16045   2020     11  PAT.DE      825.903071
16046   2020     12  PAT.DE     1015.523671

[16047 rows x 4 columns]
```

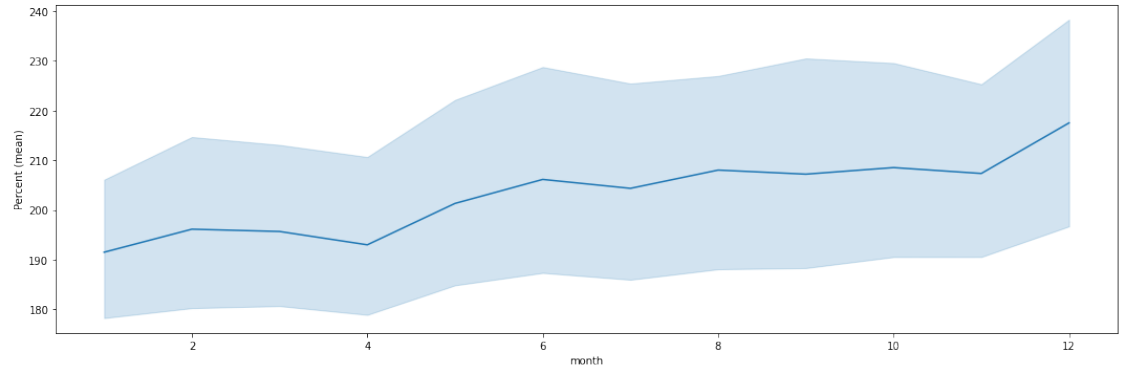
[3]:
```
plot(x=Column.MONTH, y=Column.PERCENT, data=df)
```

```
       Percent (mean)
month
1          191.522927
2          196.148531
3          195.688931
4          193.007713
5          201.329265
```


Percent (mean)
month


Percent (mean)
month


Percent (mean)
month


Percent (mean)
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
```

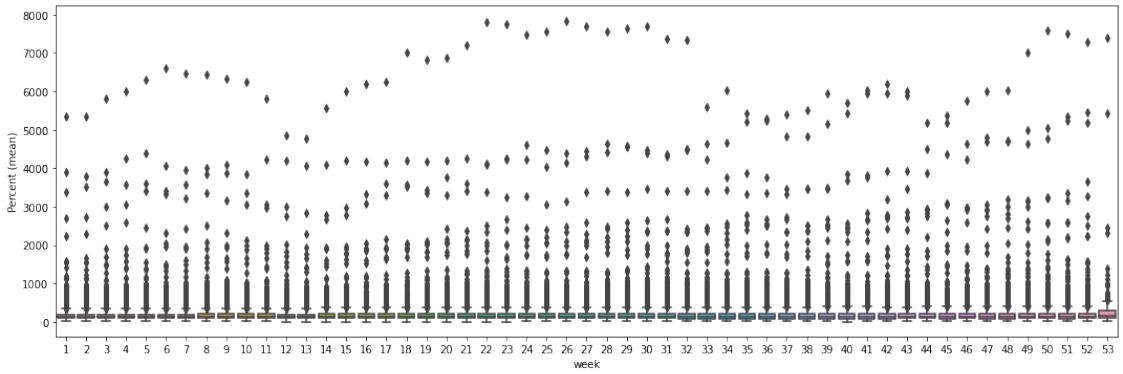
[4]:

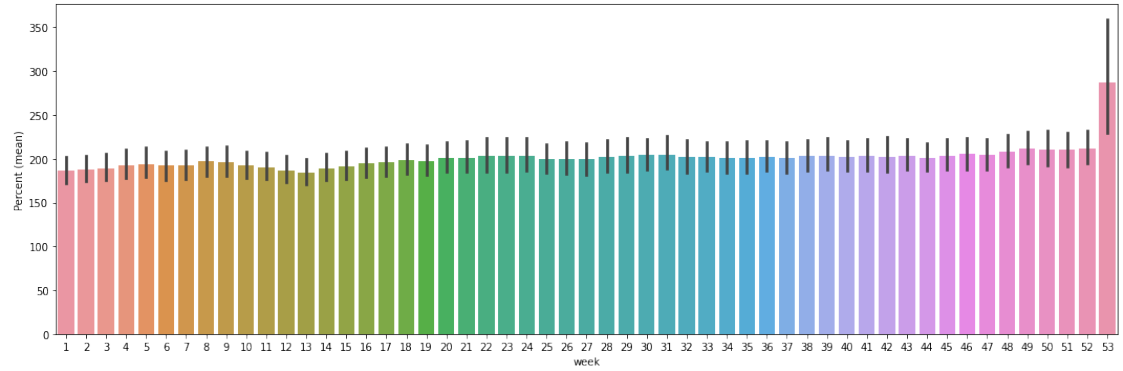
	year	week	Symbol	Percent (mean)
0	2011	1	AFX.DE	100.0
1	2011	2	AFX.DE	99.441536
2	2011	3	AFX.DE	96.684123
3	2011	4	AFX.DE	97.033157
4	2011	5	AFX.DE	98.568936
…	…	…	…	…
69079	2020	49	S92.DE	70.412096
69080	2020	50	S92.DE	67.503211
69081	2020	51	S92.DE	72.793385
69082	2020	52	S92.DE	76.358194
69083	2020	53	S92.DE	80.136893

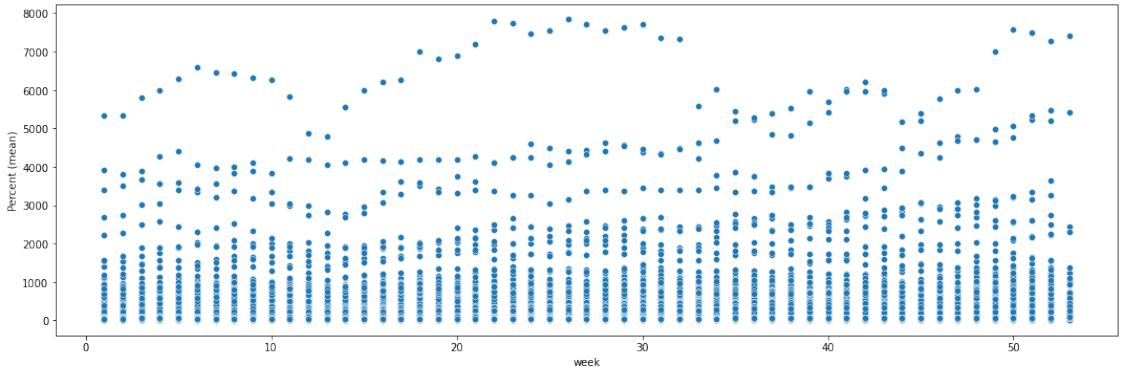
[69084 rows x 4 columns]

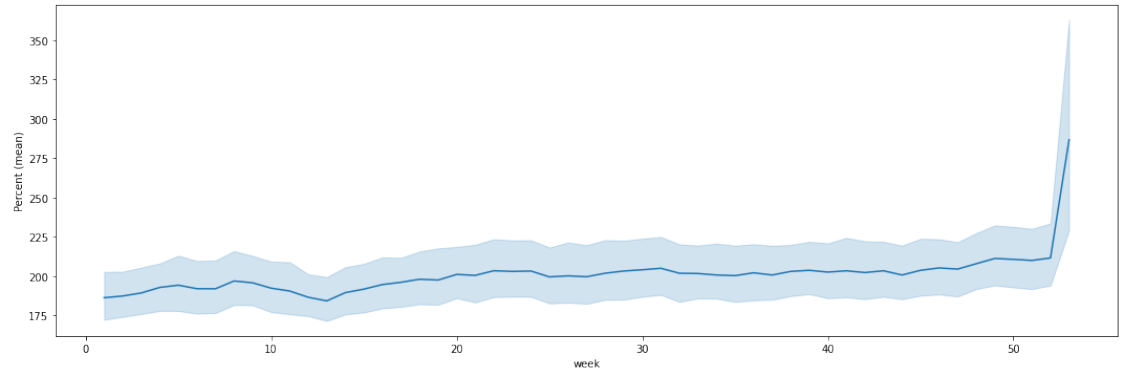
[5]:
```
plot(x=Column.WEEK, y=Column.PERCENT, data=df)
```

week	Percent (mean)
1	186.314433
2	187.405626
3	189.34534
4	192.856904
5	194.280483


Percent (mean)
week


Percent (mean)
week


Percent (mean)
week


Percent (mean)
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
```

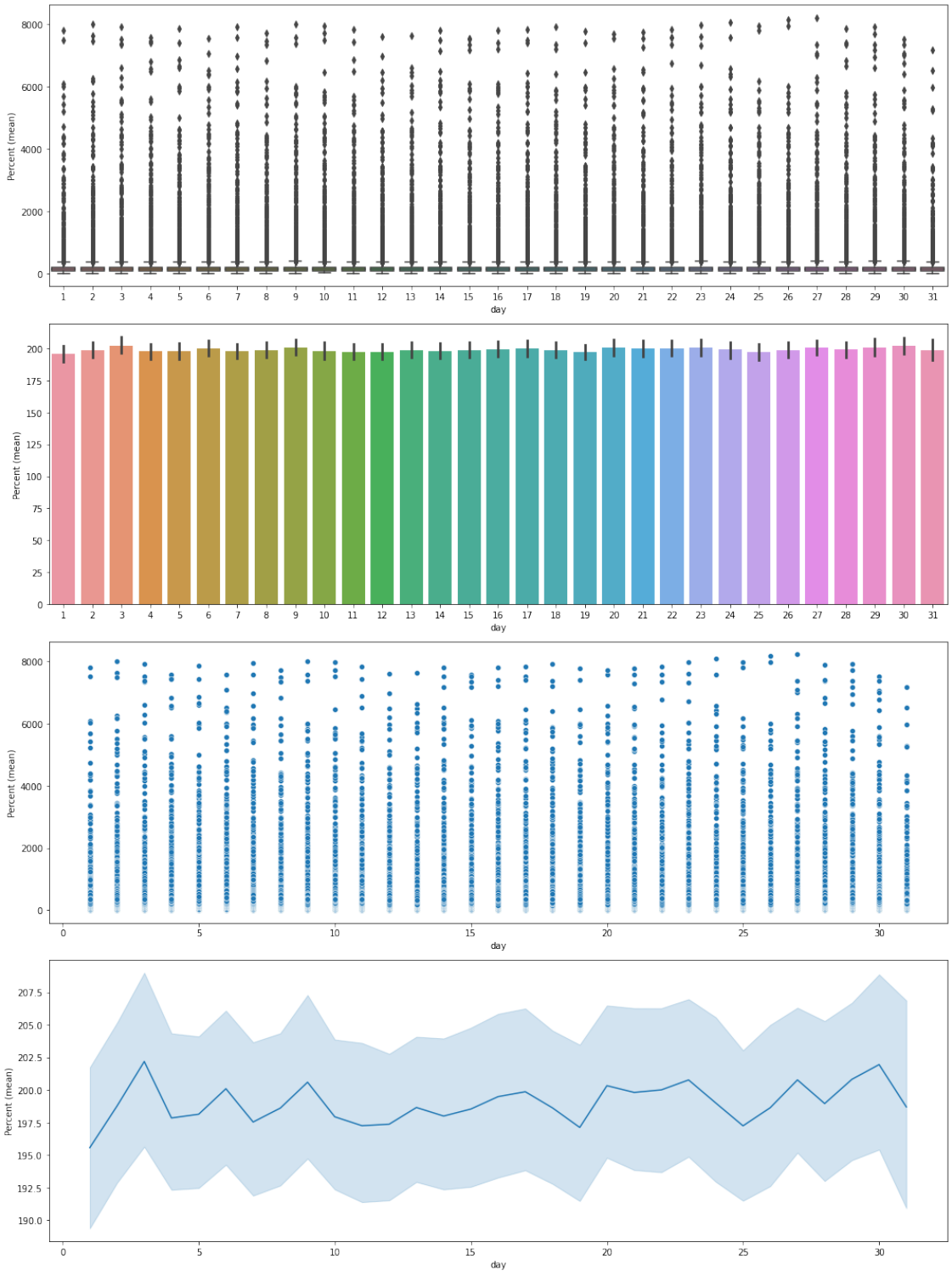
[6]:

	year	month	day	Symbol	Percent (mean)
0	2011	1	3	HYQ.DE	100.0
1	2011	1	4	HYQ.DE	100.308546
2	2011	1	5	HYQ.DE	102.989681
3	2011	1	6	HYQ.DE	102.989681
4	2011	1	7	HYQ.DE	108.628576
…	…	…	…	…	…
337615	2020	12	22	SBS.DE	377.311194
337616	2020	12	23	SBS.DE	379.191478
337617	2020	12	28	SBS.DE	379.818231
337618	2020	12	29	SBS.DE	381.071761
337619	2020	12	30	SBS.DE	379.818231

[337620 rows x 5 columns]

[7]:
```
plot(x=Column.DAY, y=Column.PERCENT, data=df)
```

day	Percent (mean)
1	195.575461
2	198.771864
3	202.190778
4	197.853051
5	198.140608


Percent (mean)
day
Percent (mean)
day
Percent (mean)
day
Percent (mean)
day

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
```

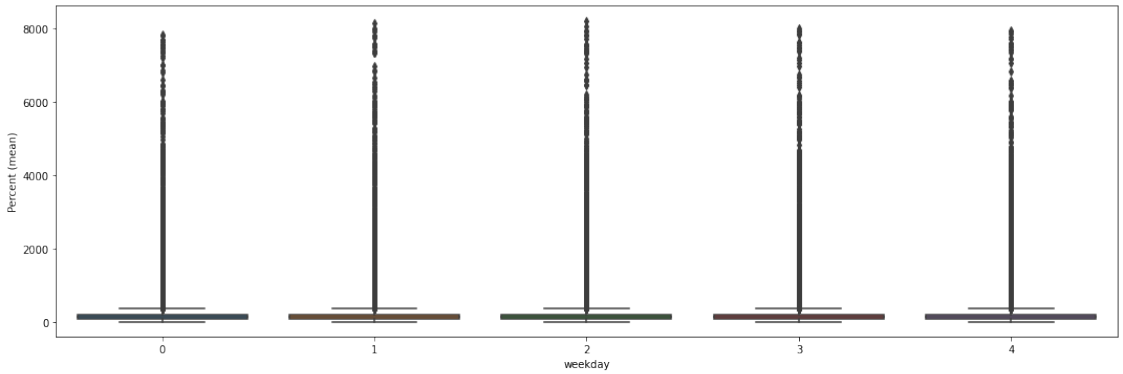
[8]:

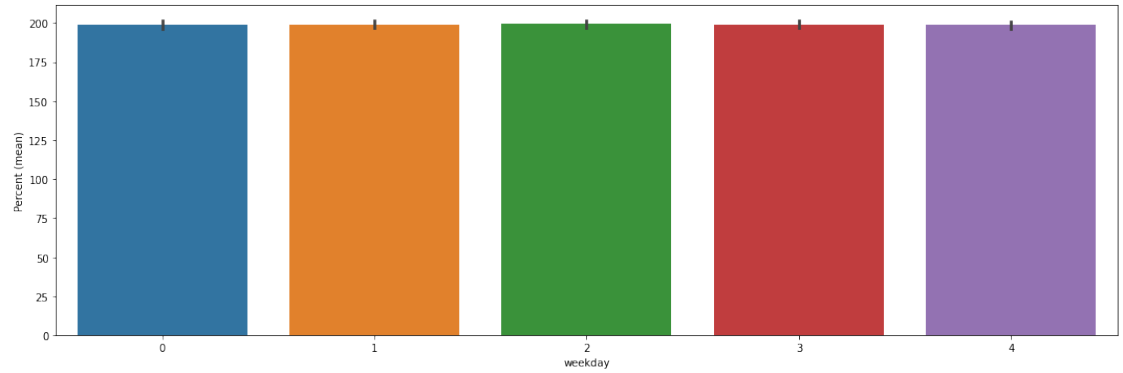
	year	week	weekday	Percent (mean)
0	2011	1	0	100.0
1	2011	1	1	100.308546
2	2011	1	2	102.989681
3	2011	1	3	102.989681
4	2011	1	4	108.628576
…	…	…	…	…
337615	2020	52	1	377.311194
337616	2020	52	2	379.191478
337617	2020	53	0	379.818231
337618	2020	53	1	381.071761
337619	2020	53	2	379.818231

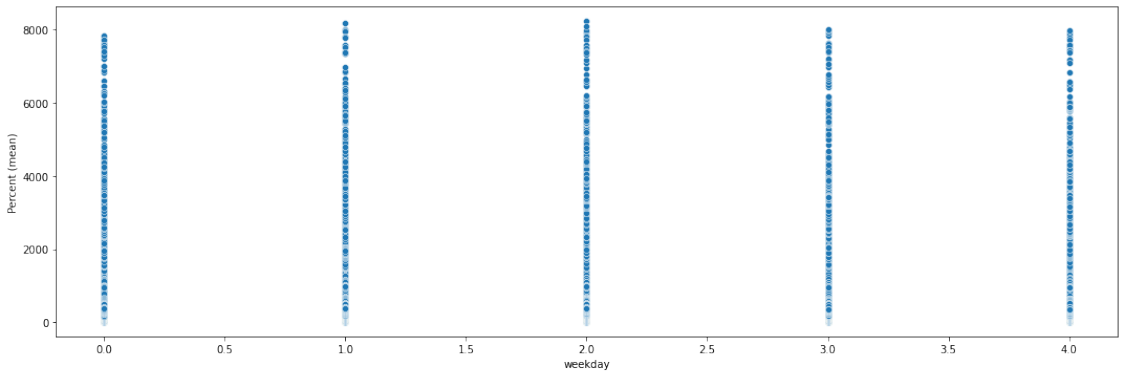
[337620 rows x 4 columns]

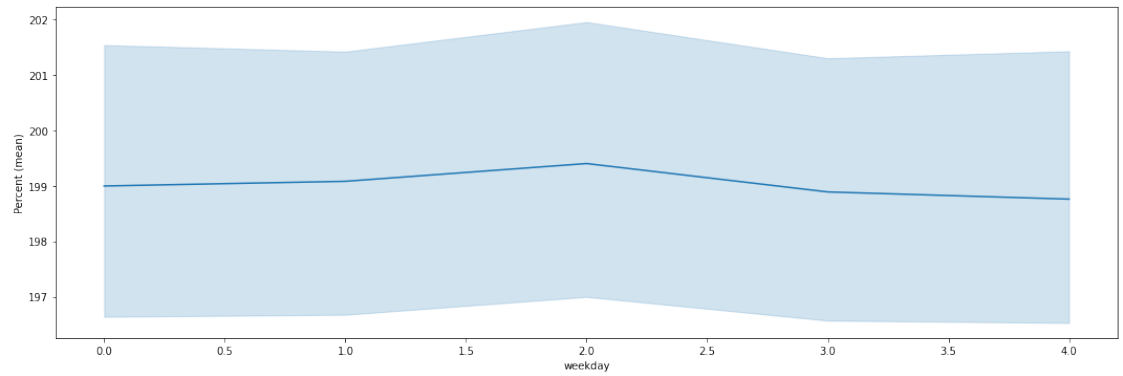
[9]:
```
plot(x=Column.WEEKDAY, y=Column.PERCENT, data=df)
```

weekday	Percent (mean)
0	198.995989
1	199.080048
2	199.400741
3	198.888945
4	198.759137


Percent (mean)
8000
6000
4000
2000
0
0
1
2
3
4
weekday


Percent (mean)
200
175
150
125
100
75
50
25
0
0
1
2
3
4
weekday


Percent (mean)
8000
6000
4000
2000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
weekday


Percent (mean)
202
201
200
199
198
197
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
```

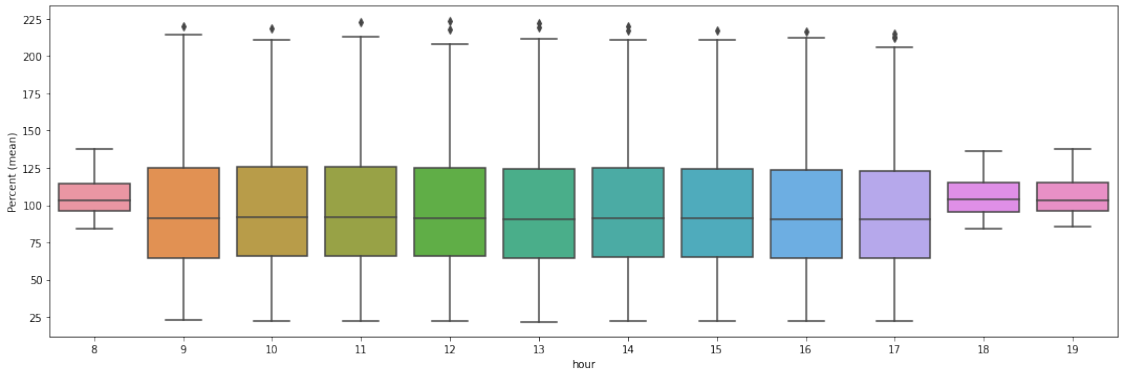
[10]:

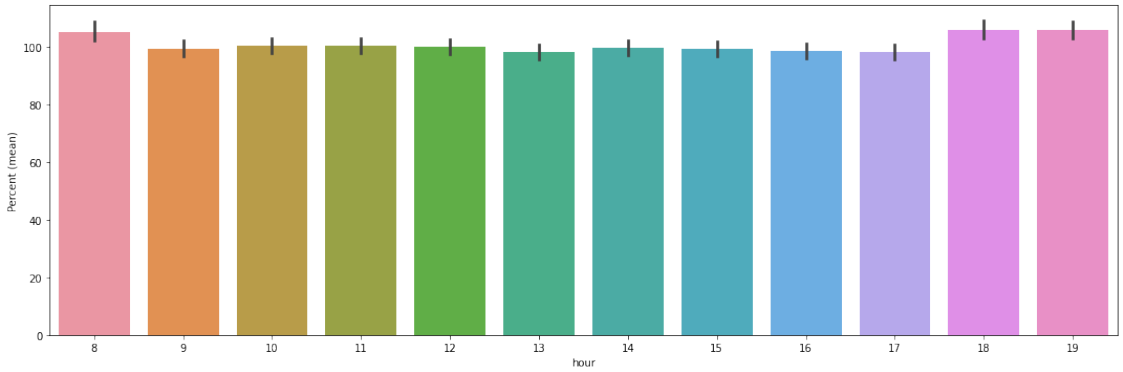
	year	week	day	hour	Symbol	Percent (mean)
0	2020	40	28	9	ENR.F	100.0
1	2020	40	28	10	ENR.F	95.592915
2	2020	40	28	11	ENR.F	102.998636
3	2020	40	28	12	ENR.F	102.998636
4	2020	40	28	13	ENR.F	99.045884
…	…	…	…	…	…	…
9843	2020	53	30	10	8TRA.DE	85.187968
9844	2020	53	30	11	8TRA.DE	86.691726
9845	2020	53	30	12	8TRA.DE	86.503759
9846	2020	53	30	13	8TRA.DE	85.563909
9847	2020	53	30	14	8TRA.DE	84.9812

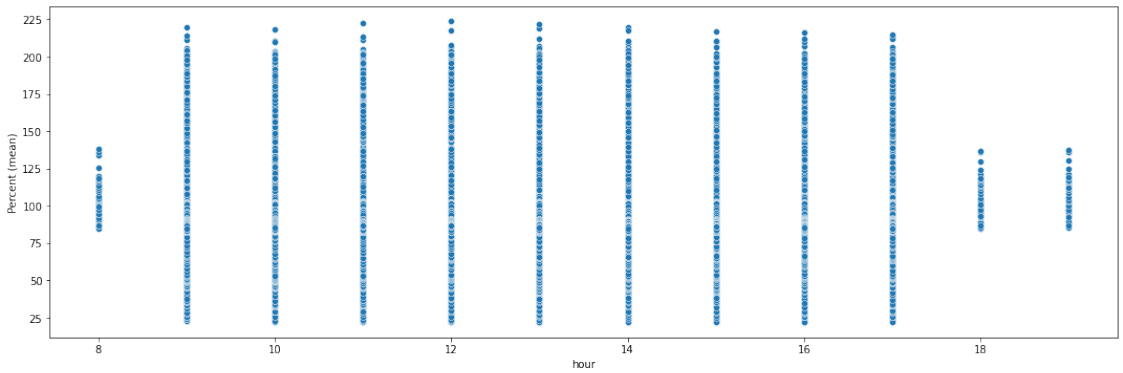
[9848 rows x 6 columns]

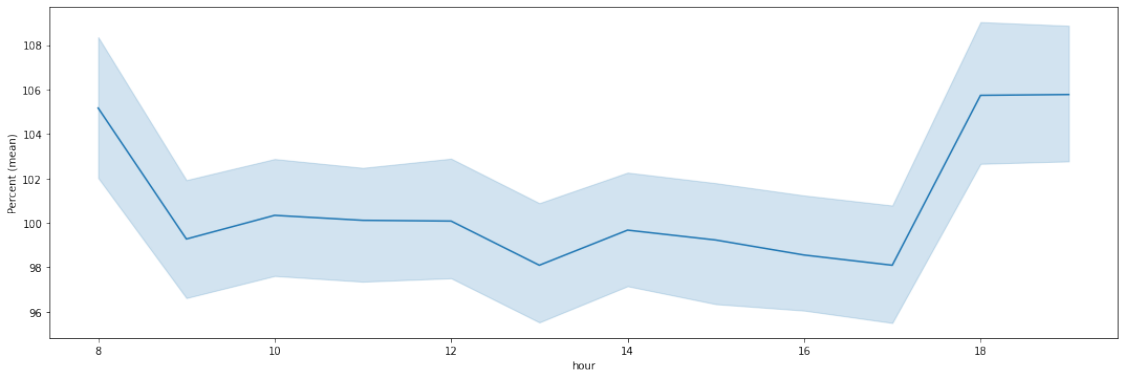
[11]:
```
plot(x=Column.HOUR, y=Column.PERCENT, data=df)
```

hour	Percent (mean)
8	105.168104
9	99.272894
10	100.338999
11	100.114165
12	100.080771


Percent (mean)
hour


Percent (mean)
hour


Percent (mean)
hour


Percent (mean)
hour

1.6 Hourly and quarterly stock price fluctuations within an day

[2]:
```
from analysis_base_first_date import get_best_time

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

df
```

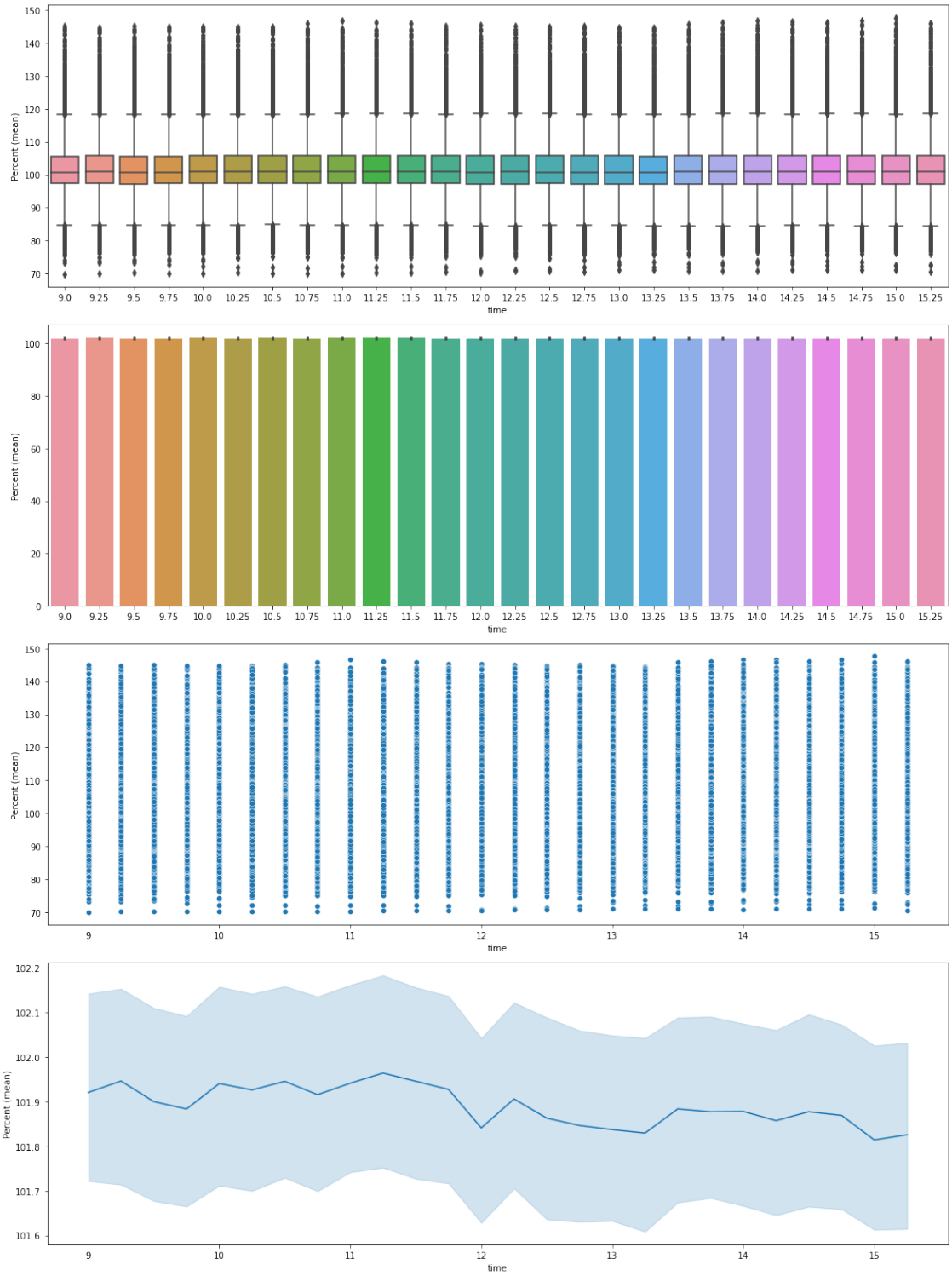
[2]:

	year	week	day	hour	minute	time	Symbol	Percent (mean)
0	2021	11	17	8	0	8.0	SZU.DE	100.0
1	2021	11	17	8	15	8.25	SZU.DE	99.782295
2	2021	11	17	8	30	8.5	SZU.DE	99.782295
3	2021	11	17	8	45	8.75	SZU.DE	99.854866
4	2021	11	17	9	0	9.0	SZU.DE	99.637154
…	…	…	…	…	…	…	…	…
207268	2021	19	13	14	15	14.25	PBB.DE	100.245593
207269	2021	19	13	14	30	14.5	PBB.DE	100.352376
207270	2021	19	13	14	45	14.75	PBB.DE	100.352376
207271	2021	19	13	15	0	15.0	PBB.DE	100.480513
207272	2021	19	13	15	15	15.25	PBB.DE	100.65136

[207273 rows x 8 columns]

[4]:
```
# NOTE: filter extreme points, plot df first and if charts are bad try with fdf
fdf = df[df[Column.TIME].isin(np.arange(9, 15.5, 0.25))].copy()

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

time	Percent (mean)
9.00	101.920529
9.25	101.946105
9.50	101.900031
9.75	101.883461
10.00	101.940452


Percent (mean)
time
9.0 9.25 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
9 10 11 12 13 14 15

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
```

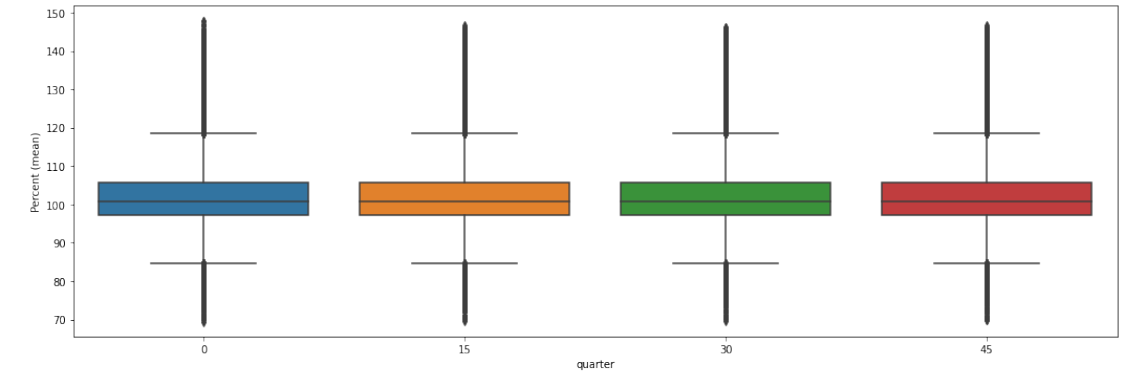
[14]:

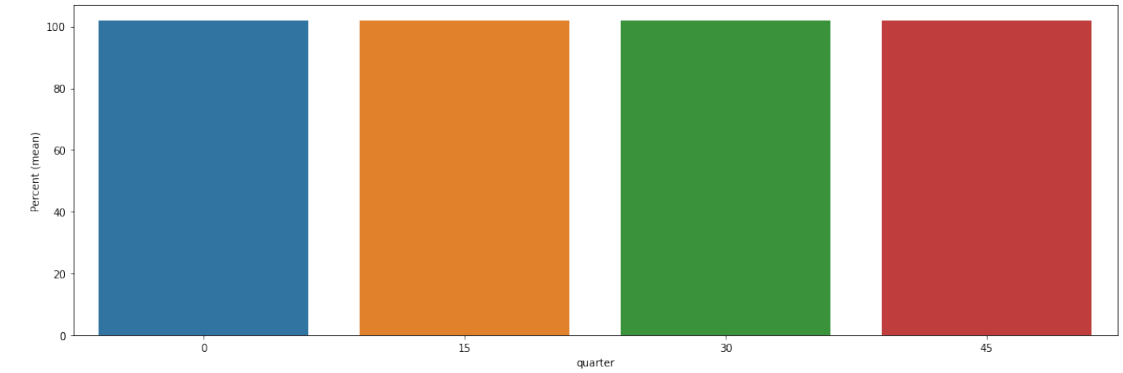
	year	week	day	hour	minute	quarter	Symbol	Percent (mean)
0	2021	11	17	8	0	0	SZU.DE	100.0
1	2021	11	17	8	15	15	SZU.DE	99.782295
2	2021	11	17	8	30	30	SZU.DE	99.782295
3	2021	11	17	8	45	45	SZU.DE	99.854866
4	2021	11	17	9	0	0	SZU.DE	99.637154
…	…	…	…	…	…	…	…	…
207268	2021	19	13	14	15	15	PBB.DE	100.245593
207269	2021	19	13	14	30	30	PBB.DE	100.352376
207270	2021	19	13	14	45	45	PBB.DE	100.352376
207271	2021	19	13	15	0	0	PBB.DE	100.480513
207272	2021	19	13	15	15	15	PBB.DE	100.65136

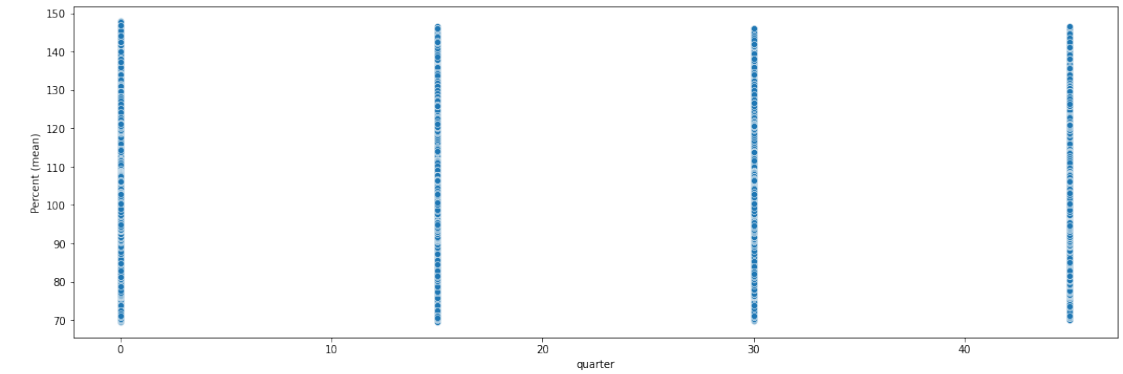
[207273 rows x 8 columns]

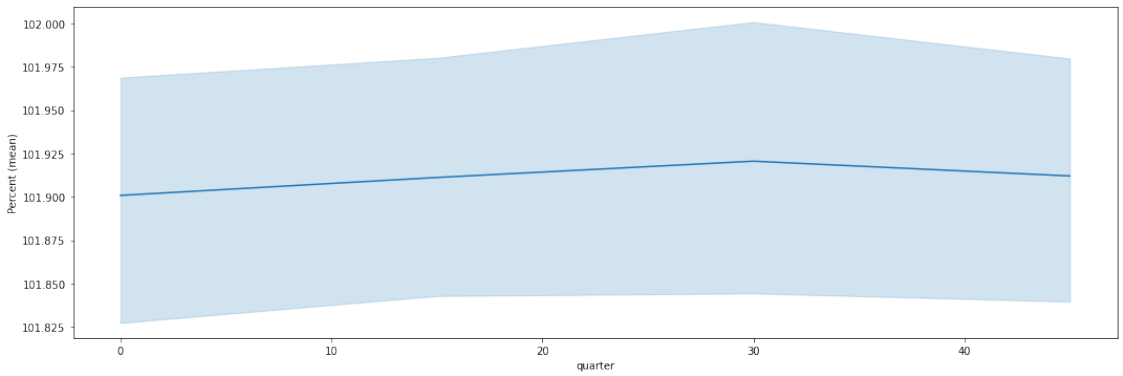
[15]:
```
plot(x=Column.QUARTER, y=Column.PERCENT, data=df)
```

quarter	Percent (mean)
0	101.900798
15	101.911116
30	101.920512
45	101.911991


Percent (mean)
quarter
0
15
30
45


Percent (mean)
quarter
0
15
30
45


Percent (mean)
quarter
0
10
20
30
40


Percent (mean)
quarter
0
10
20
30
40

[ ]: