analysis_base_first_date

May 17, 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_20, limit=LIMIT)
df
```

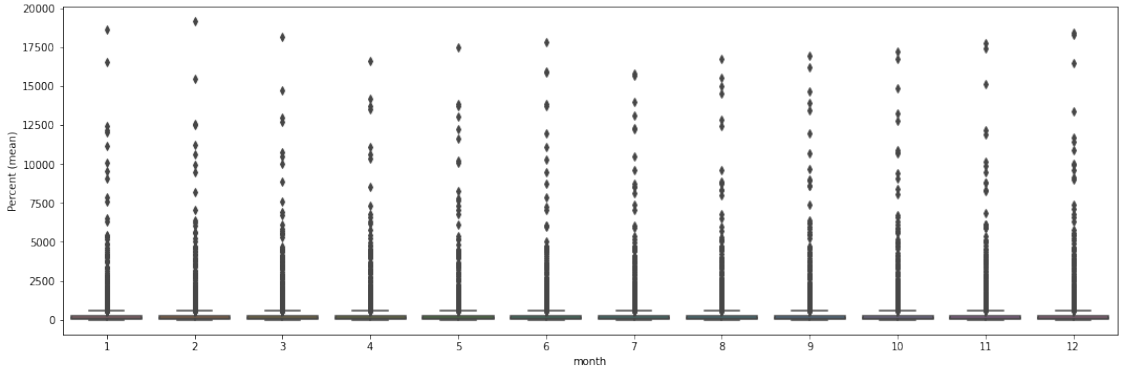
[2]:

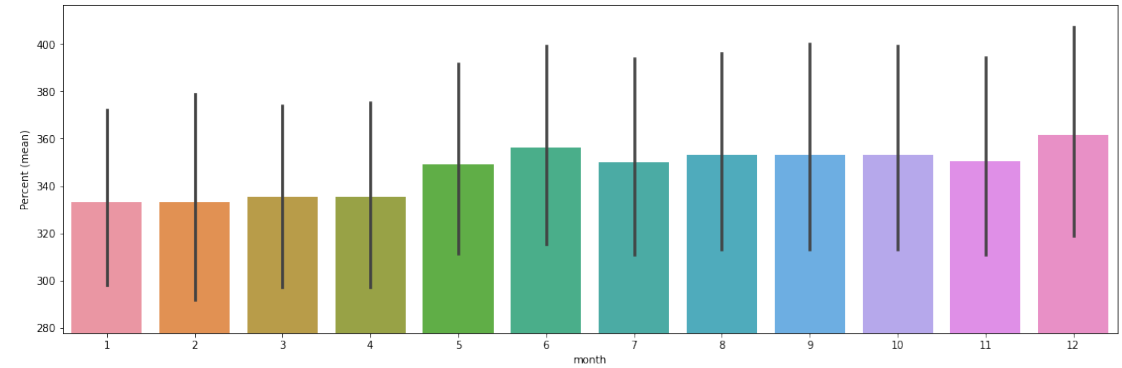
	year	month	Symbol	Percent (mean)
0	2009	2	PAH3.DE	100.0

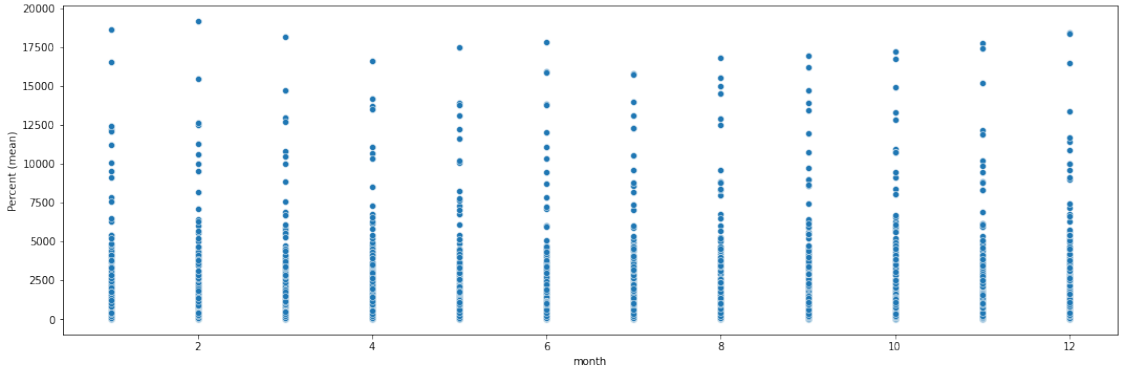
```
1       2009      3  PAH3.DE       73.903007
2       2009      4  PAH3.DE       81.986145
3       2009      5  PAH3.DE      126.073912
4       2009      6  PAH3.DE      101.039264
…       …     …       …              …
28237   2020      8  SRT3.DE    15523.810229
28238   2020      9  SRT3.DE    16952.381722
28239   2020     10  SRT3.DE    16752.381132
28240   2020     11  SRT3.DE    17409.524891
28241   2020     12  SRT3.DE    18333.334166

[28242 rows x 4 columns]
```

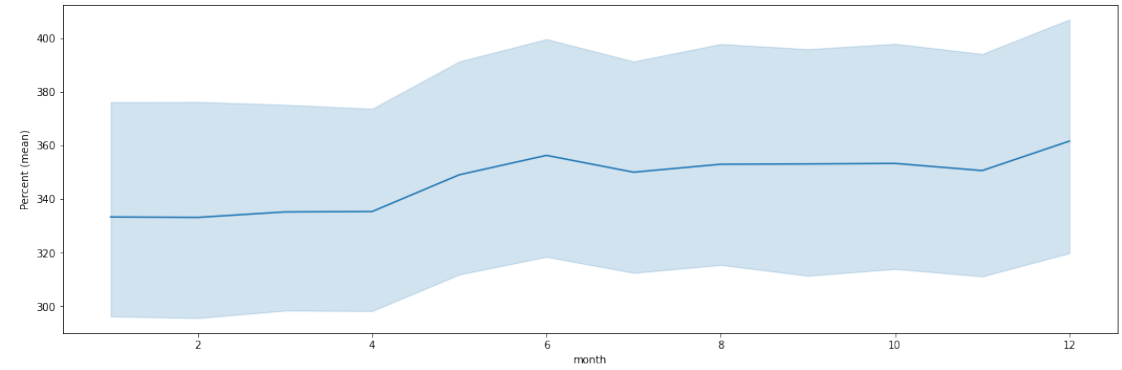
[3]:
```
plot(x=Column.MONTH, y=Column.PERCENT, data=df)
```

```
       Percent (mean)
month
1          333.315473
2          333.164385
3          335.206428
4          335.354506
5          349.072129
```


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_20, limit=LIMIT)

df
```

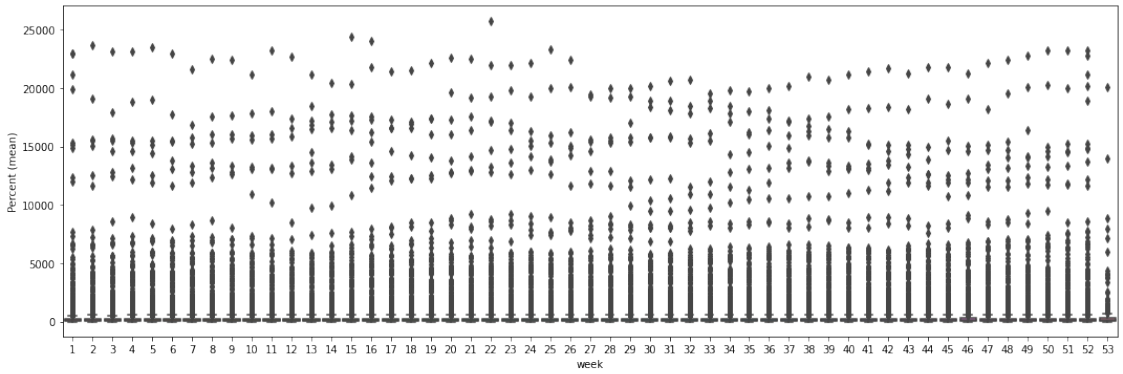
[4]:

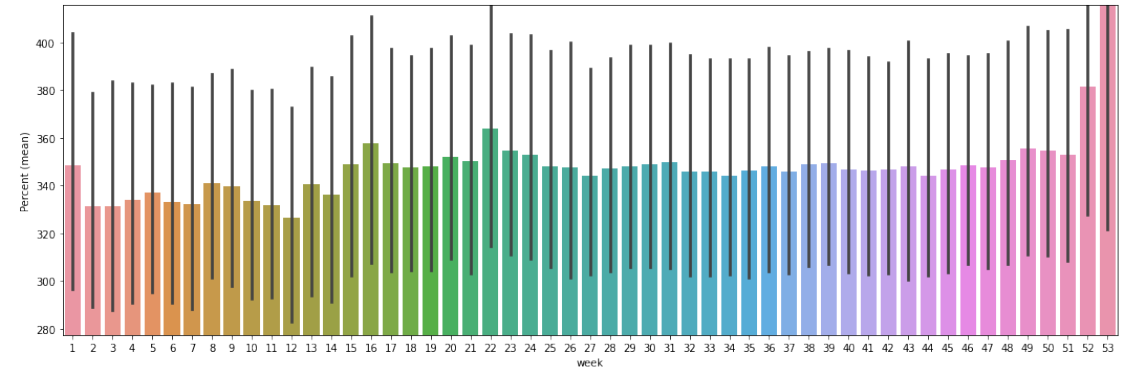
	year	week	Symbol	Percent (mean)
0	2001	1	ALV.DE	100.0
1	2001	2	ALV.DE	94.004016
2	2001	3	ALV.DE	90.689911
3	2001	4	ALV.DE	88.283991
4	2001	5	ALV.DE	90.378825
…	…	…	…	…
116536	2020	49	O2D.DE	50.668306
116537	2020	50	O2D.DE	50.668306
116538	2020	51	O2D.DE	48.348758
116539	2020	52	O2D.DE	48.944603
116540	2020	53	O2D.DE	49.051007

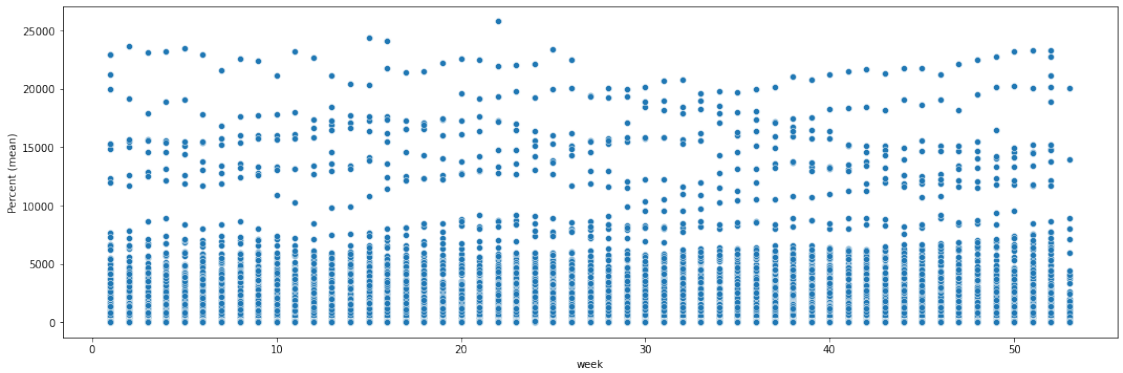
[116541 rows x 4 columns]

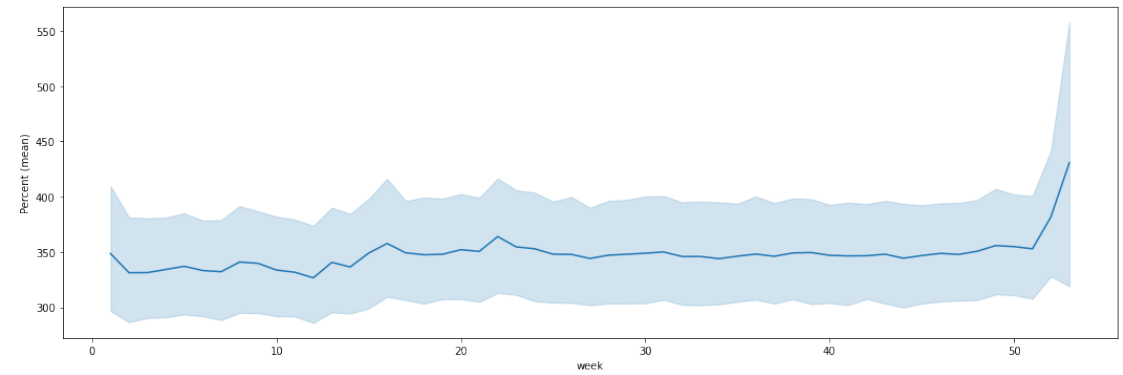
[5]:
```
plot(x=Column.WEEK, y=Column.PERCENT, data=df)
```

week	Percent (mean)
1	348.525357
2	331.262644
3	331.326693
4	334.150612
5	337.016489


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_20, limit=LIMIT)

df
```

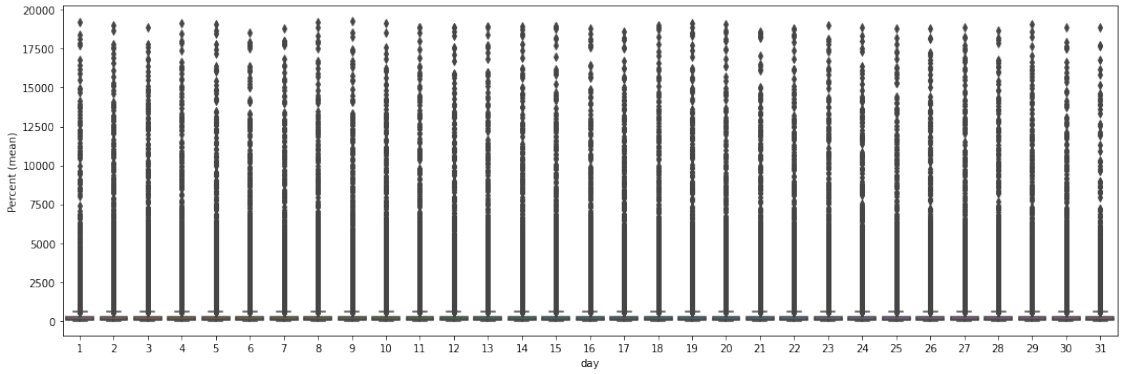
[6]:

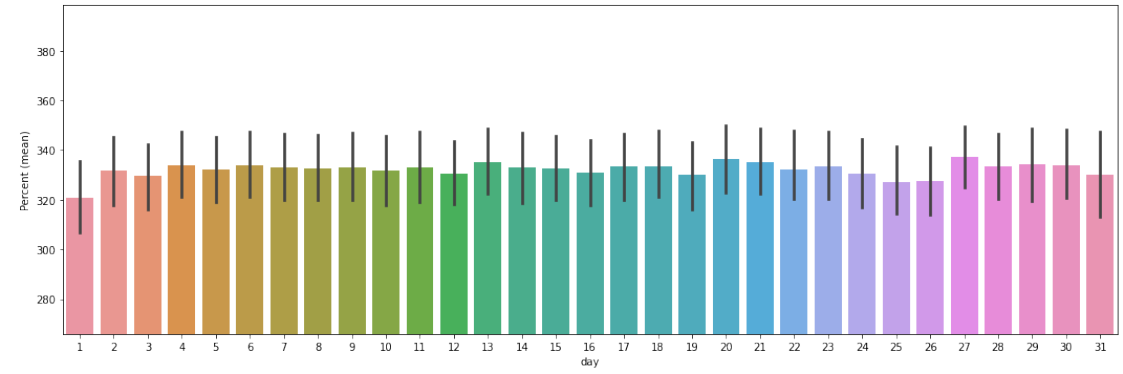
	year	month	day	Symbol	Percent (mean)
0	2007	4	3	AOX.DE	100.0
1	2007	4	4	AOX.DE	101.226999
2	2007	4	5	AOX.DE	102.453997
3	2007	4	10	AOX.DE	102.392646
4	2007	4	11	AOX.DE	101.226999
…	…	…	…	…	…
591116	2020	12	22	SBS.DE	2574.839239
591117	2020	12	23	SBS.DE	2587.670631
591118	2020	12	28	SBS.DE	2591.947707
591119	2020	12	29	SBS.DE	2600.502023
591120	2020	12	30	SBS.DE	2591.947707

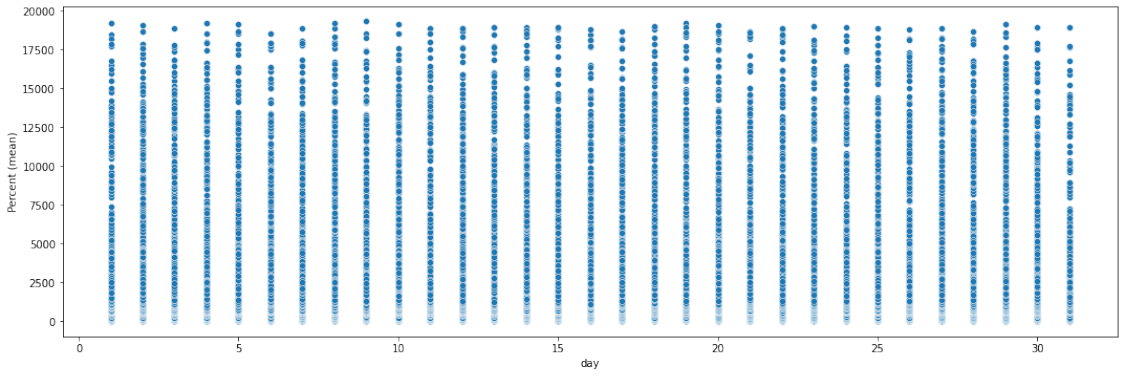
[591121 rows x 5 columns]

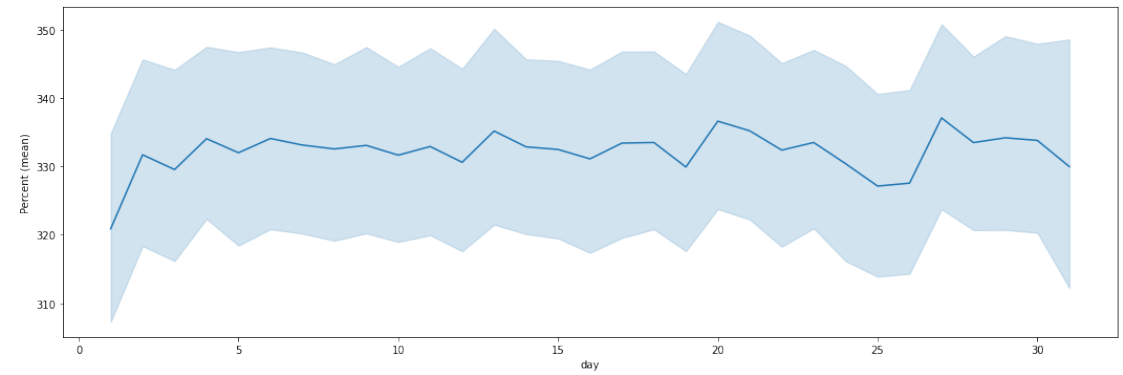
[7]:
```
plot(x=Column.DAY, y=Column.PERCENT, data=df)
```

day	Percent (mean)
1	320.863345
2	331.684501
3	329.527187
4	334.033718
5	331.985347


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_20, limit=LIMIT)

df
```

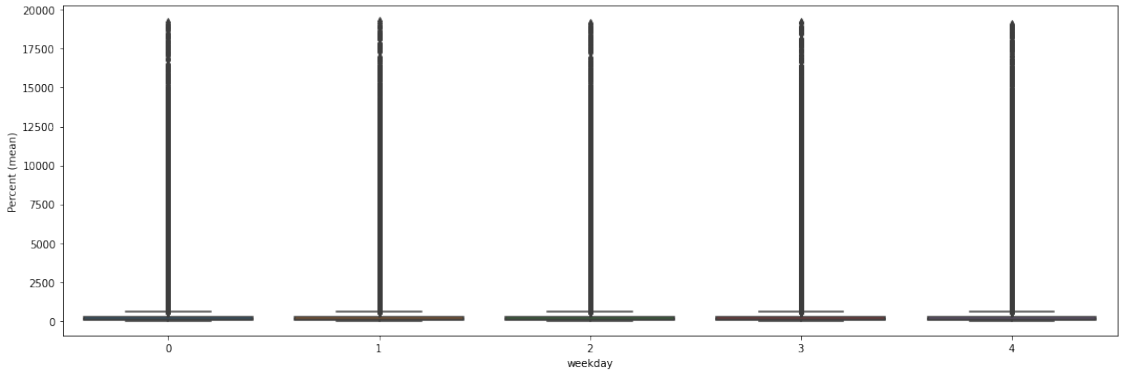
[8]:

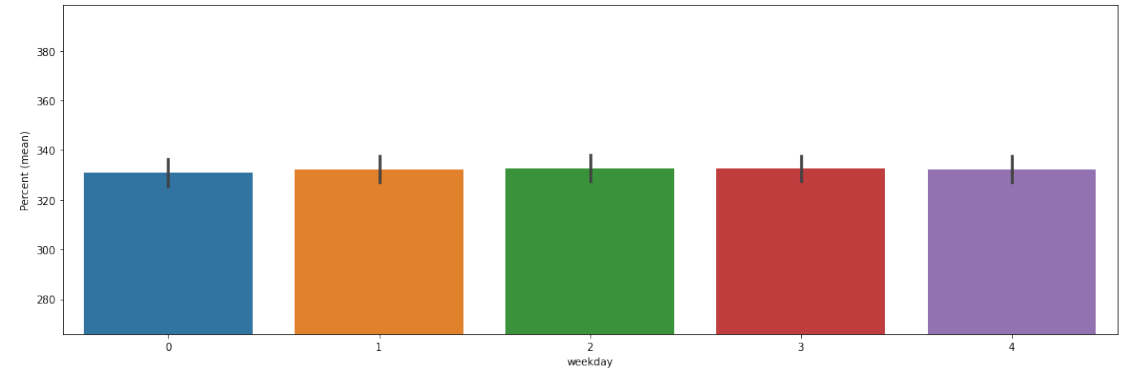
	year	week	weekday	Percent (mean)
0	2007	14	1	100.0
1	2007	14	2	101.226999
2	2007	14	3	102.453997
3	2007	15	1	102.392646
4	2007	15	2	101.226999
…	…	…	…	…
591116	2020	52	1	2574.839239
591117	2020	52	2	2587.670631
591118	2020	53	0	2591.947707
591119	2020	53	1	2600.502023
591120	2020	53	2	2591.947707

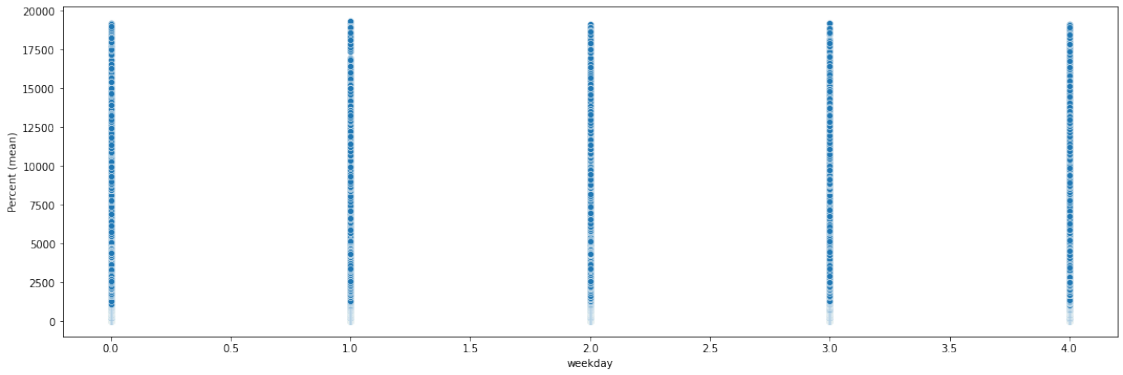
[591121 rows x 4 columns]

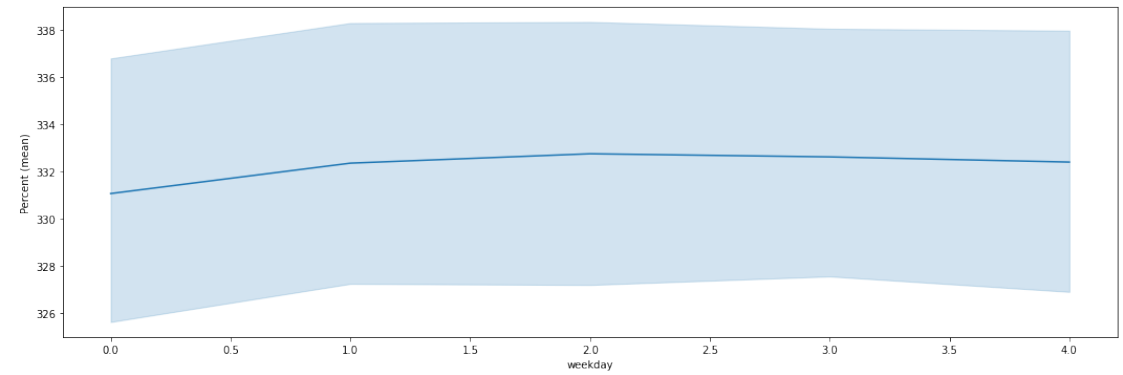
[9]:
```
plot(x=Column.WEEKDAY, y=Column.PERCENT, data=df)
```

weekday	Percent (mean)
0	331.056053
1	332.344396
2	332.742214
3	332.609612
4	332.390242


Percent (mean)
weekday


Percent (mean)
weekday


Percent (mean)
weekday


Percent (mean)
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
```

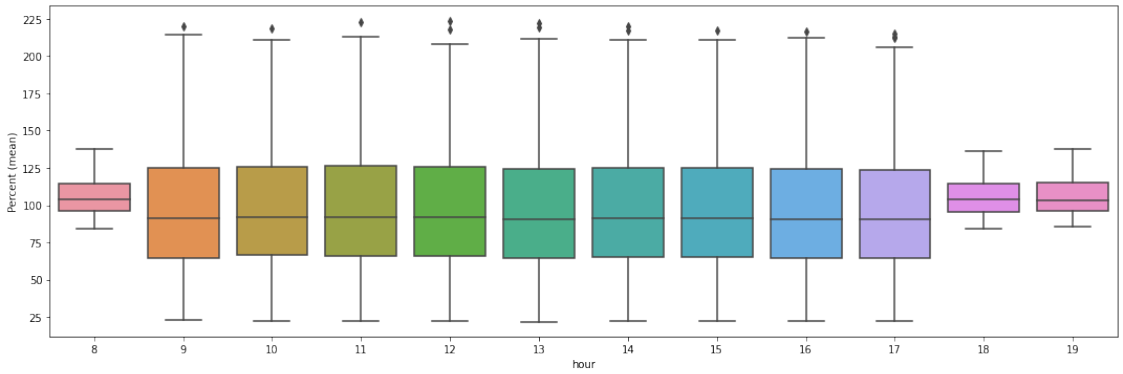
[10]:

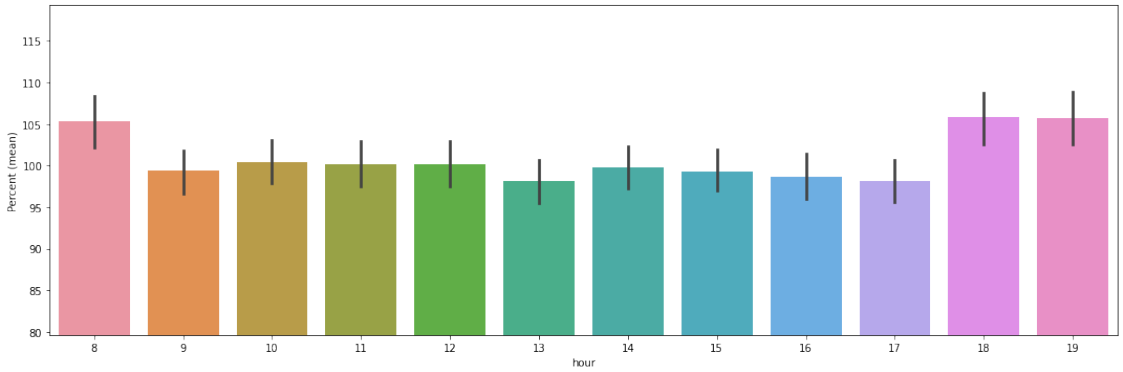
	year	week	day	hour	Symbol	Percent (mean)
0	2019	27	1	9	8TRA.DE	100.0
1	2019	27	1	10	8TRA.DE	100.883454
2	2019	27	1	11	8TRA.DE	100.469925
3	2019	27	1	12	8TRA.DE	100.977444
4	2019	27	1	13	8TRA.DE	101.165412
…	…	…	…	…	…	…
9881	2020	53	30	10	GFG.DE	218.545873
9882	2020	53	30	11	GFG.DE	222.818803
9883	2020	53	30	12	GFG.DE	223.646537
9884	2020	53	30	13	GFG.DE	221.968689
9885	2020	53	30	14	GFG.DE	217.4273

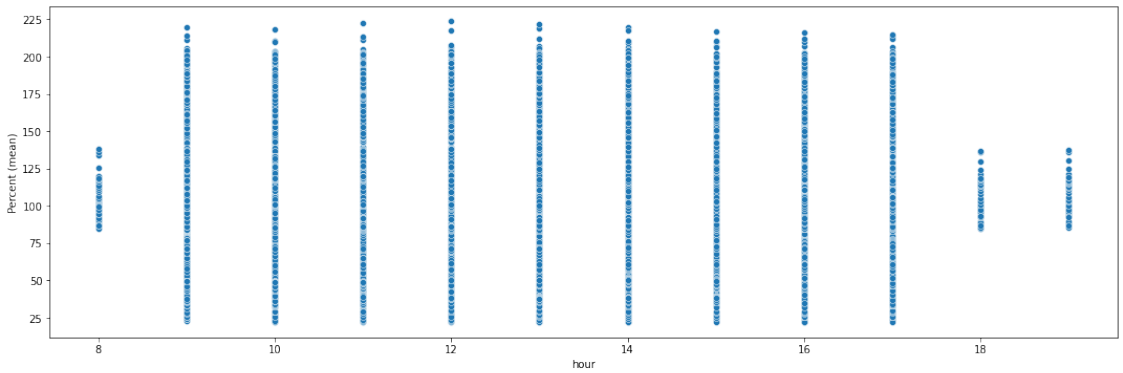
[9886 rows x 6 columns]

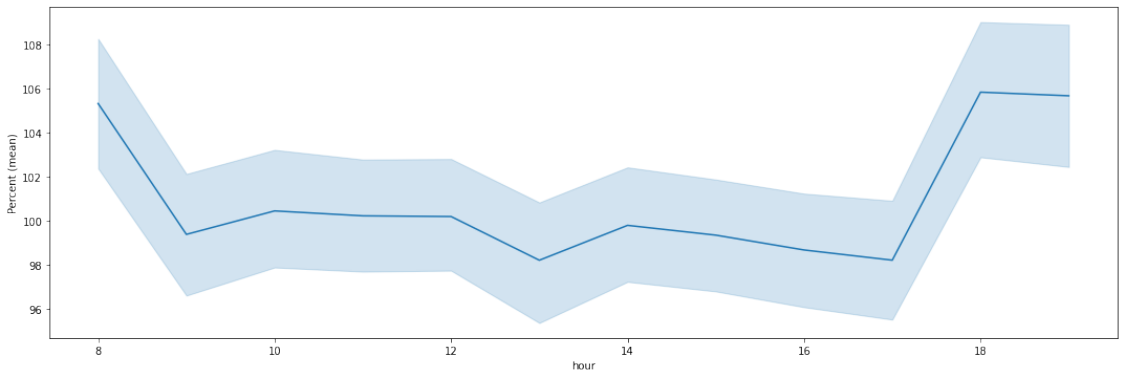
[11]:
```
plot(x=Column.HOUR, y=Column.PERCENT, data=df)
```

hour	Percent (mean)
8	105.315065
9	99.380372
10	100.445305
11	100.221499
12	100.189718


Percent (mean)
hour


Percent (mean)
hour


Percent (mean)
hour


Percent (mean)
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
```

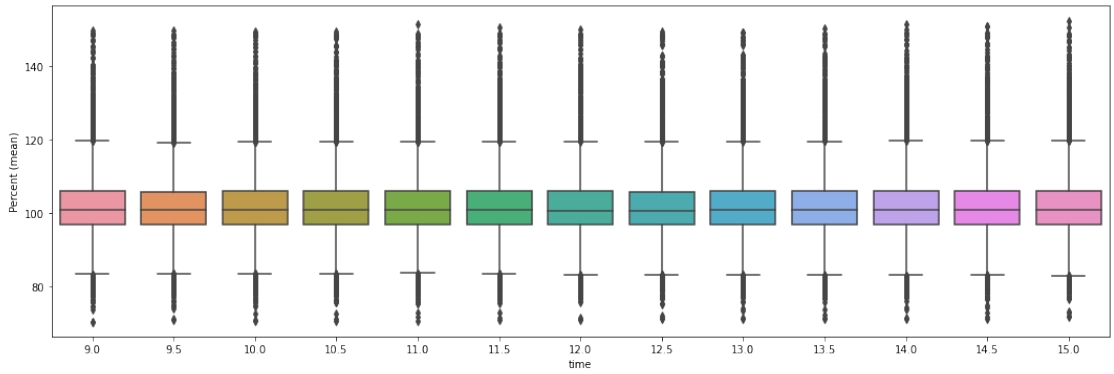
[12]:

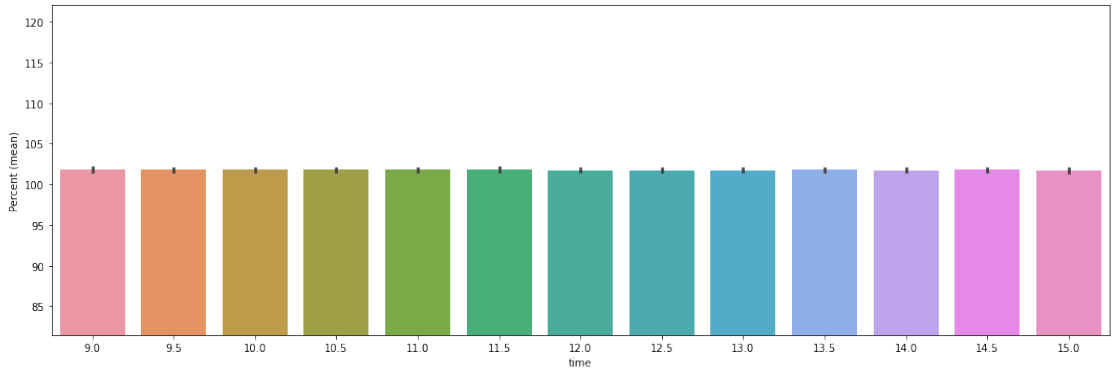
	year	week	day	hour	minute	time	Symbol	Percent (mean)
0	2021	11	18	8	0	8.0	NDX1.DE	100.0
1	2021	11	18	8	30	8.5	NDX1.DE	99.91103
2	2021	11	18	9	0	9.0	NDX1.DE	98.220642
3	2021	11	18	9	30	9.5	NDX1.DE	98.309612
4	2021	11	18	10	0	10.0	NDX1.DE	98.576514
…	…	…	…	…	…	…	…	…
107084	2021	19	14	13	0	13.0	EVK.DE	100.135688
107085	2021	19	14	13	30	13.5	EVK.DE	100.305292
107086	2021	19	14	14	0	14.0	EVK.DE	100.576662
107087	2021	19	14	14	30	14.5	EVK.DE	100.305292
107088	2021	19	14	15	0	15.0	EVK.DE	100.203532

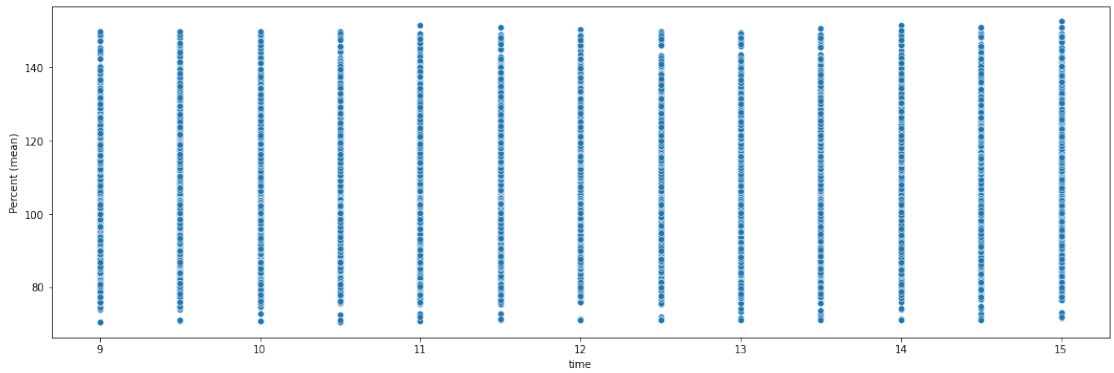
[107089 rows x 8 columns]

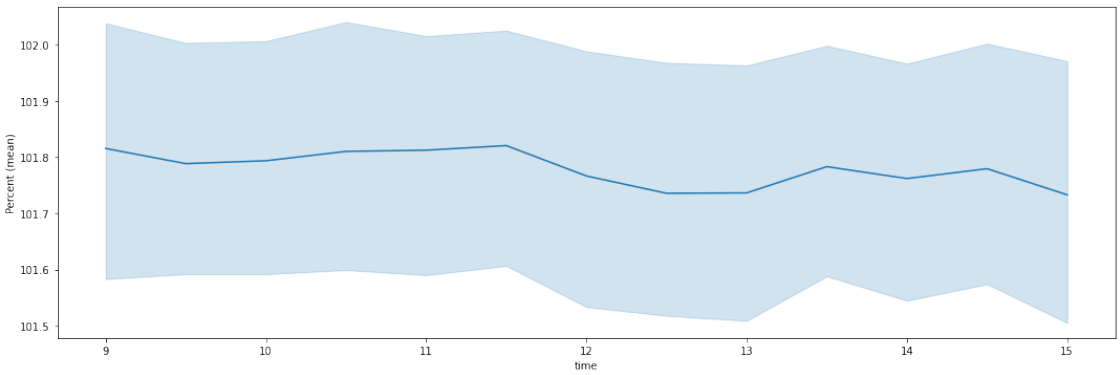
[13]:
```
# NOTE: filter extreme points, plot df first and if charts are bad try with fdf
fdf = df[df[Column.TIME].isin(np.arange(9, 15.5, 0.25))].copy()

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

time	Percent (mean)
9.0	101.815371
9.5	101.788393
10.0	101.793488
10.5	101.810181
11.0	101.812312


Percent (mean)
140
120
100
80
9.0
9.5
10.0
10.5
11.0
11.5
12.0
12.5
13.0
13.5
14.0
14.5
15.0
time


Percent (mean)
120
115
110
105
100
95
90
85
9.0
9.5
10.0
10.5
11.0
11.5
12.0
12.5
13.0
13.5
14.0
14.5
15.0
time


Percent (mean)
140
120
100
80
9
10
11
12
13
14
15
time


Percent (mean)
102.0
101.9
101.8
101.7
101.6
101.5
9
10
11
12
13
14
15
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
```

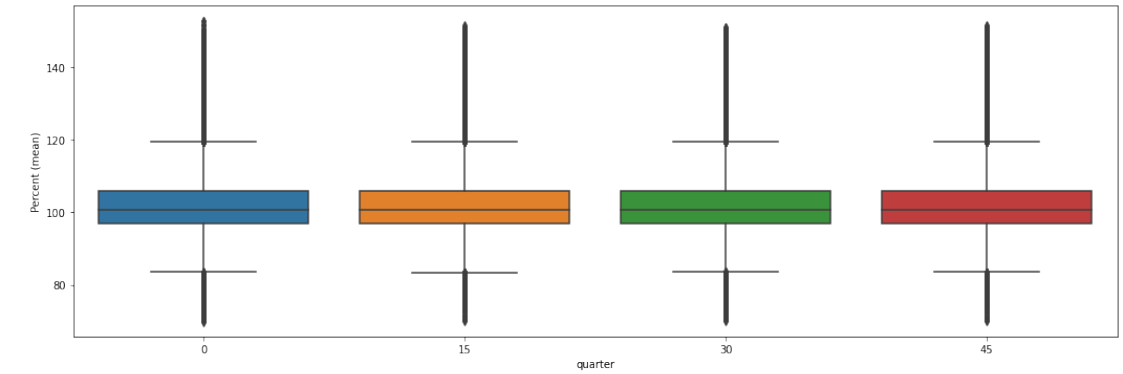
[14]:

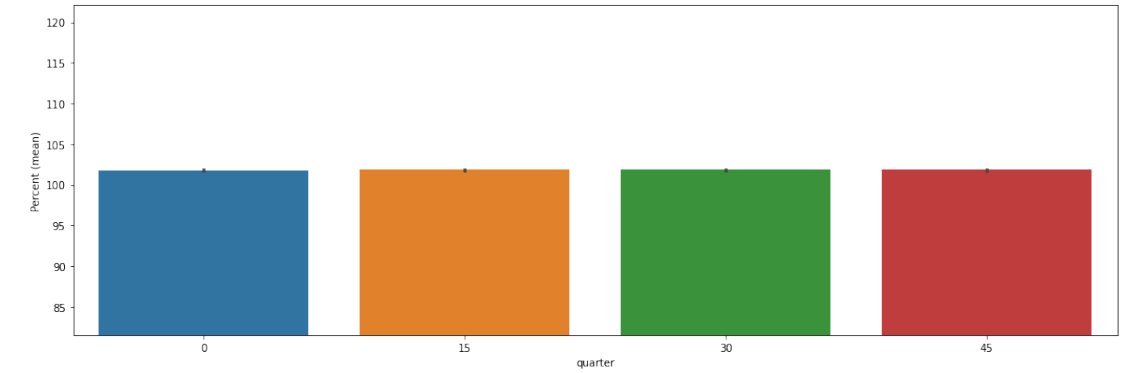
	year	week	day	hour	minute	quarter	Symbol	Percent (mean)
0	2021	11	18	8	0	0	NDX1.DE	100.0
1	2021	11	18	8	15	15	NDX1.DE	100.0
2	2021	11	18	8	30	30	NDX1.DE	99.91103
3	2021	11	18	8	45	45	NDX1.DE	99.110325
4	2021	11	18	9	0	0	NDX1.DE	98.220642
…	…	…	…	…	…	…	…	…
191087	2021	19	12	13	45	45	SBS.DE	91.03215
191088	2021	19	12	14	0	0	SBS.DE	90.862948
191089	2021	19	12	14	30	30	SBS.DE	90.524537
191090	2021	19	12	15	0	0	SBS.DE	90.355335
191091	2021	19	12	15	15	15	SBS.DE	90.693739

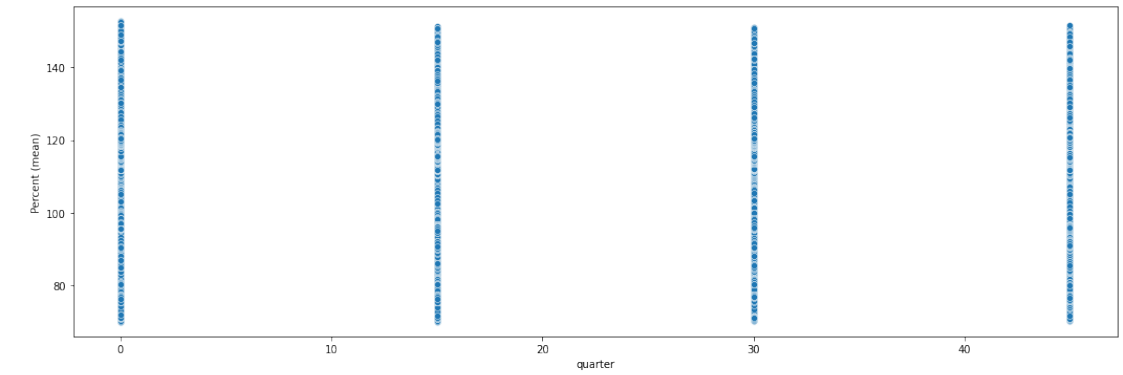
[191092 rows x 8 columns]

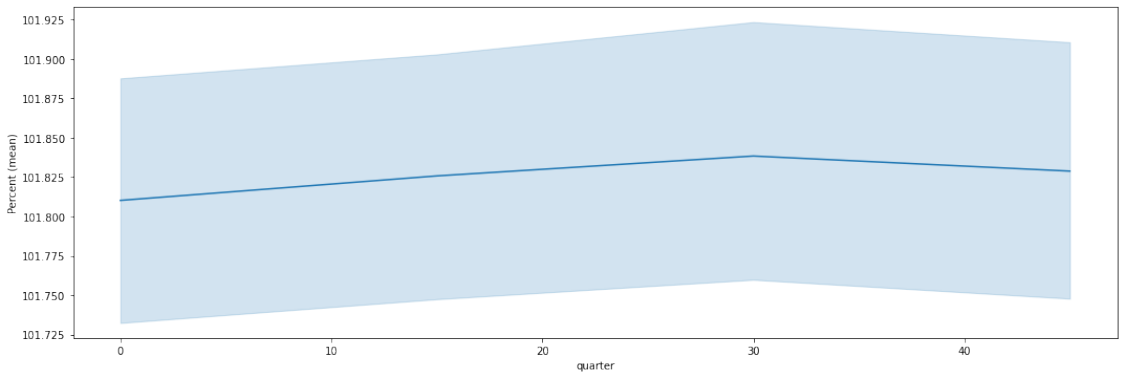
[15]:
```
plot(x=Column.QUARTER, y=Column.PERCENT, data=df)
```

quarter	Percent (mean)
0	101.810201
15	101.825754
30	101.838374
45	101.828805


Percent (mean)
140
120
100
80
0
15
30
45
quarter


Percent (mean)
120
115
110
105
100
95
90
85
0
15
30
45
quarter


Percent (mean)
140
120
100
80
0
10
20
30
40
quarter


Percent (mean)
101.925
101.900
101.875
101.850
101.825
101.800
101.775
101.750
101.725
0
10
20
30
40
quarter

[ ]: