analysis

May 17, 2021

1 Analysis of stock prices in different time periods

NOTE: `base date point` will be set separatly for each period.

Example: if we want to get daily prices within a week then each Monday will be set as `base date point`

[1]:
```
import sys

sys.path.append('..')

from analysis 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 import get_best_month

df = get_best_month(FILENAME, YahooRange.YEARS_20, limit=LIMIT)
df
```

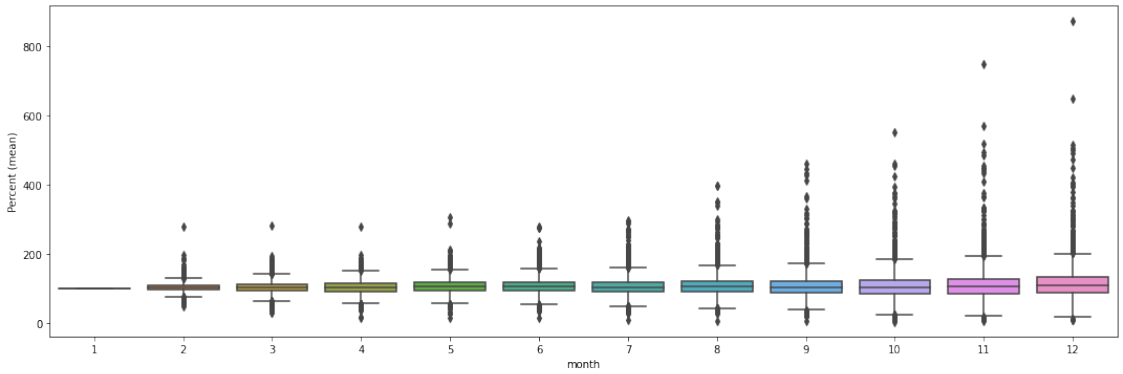
[2]:

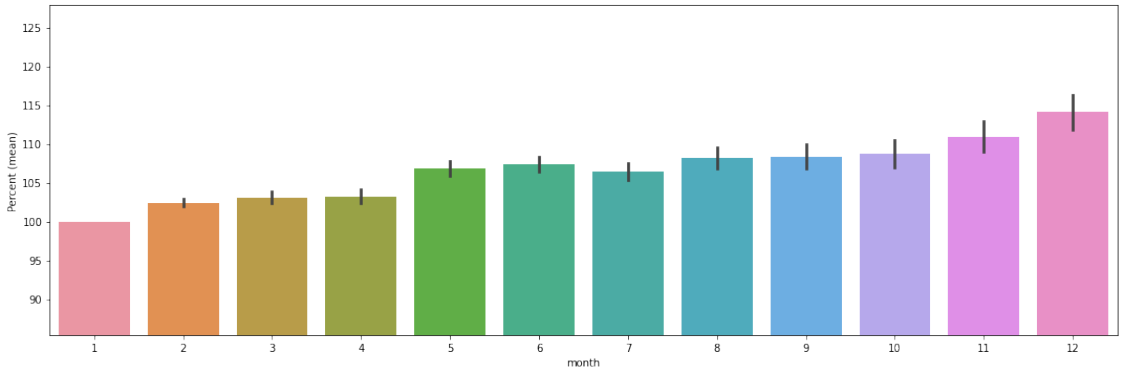
	year	month	Symbol	Percent (mean)
0	2010	1	PAH3.DE	100.0

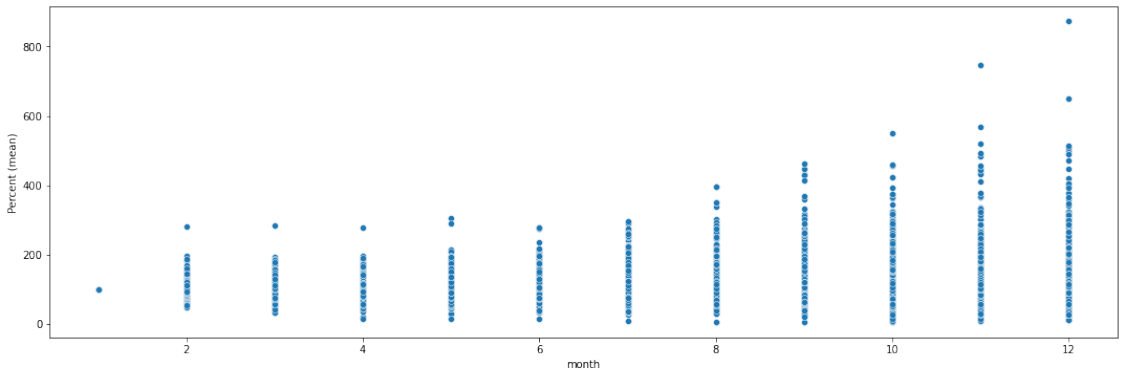
```
1      2010      2  PAH3.DE       93.392781
2      2010      3  PAH3.DE       84.830821
3      2010      4  PAH3.DE      104.252405
4      2010      5  PAH3.DE       99.908553
…       …       …       …              …
27727  2020      8  SRT3.DE      170.859536
27728  2020      9  SRT3.DE      186.582806
27729  2020     10  SRT3.DE      184.381542
27730  2020     11  SRT3.DE      191.614256
27731  2020     12  SRT3.DE      201.781967

[27732 rows x 4 columns]
```

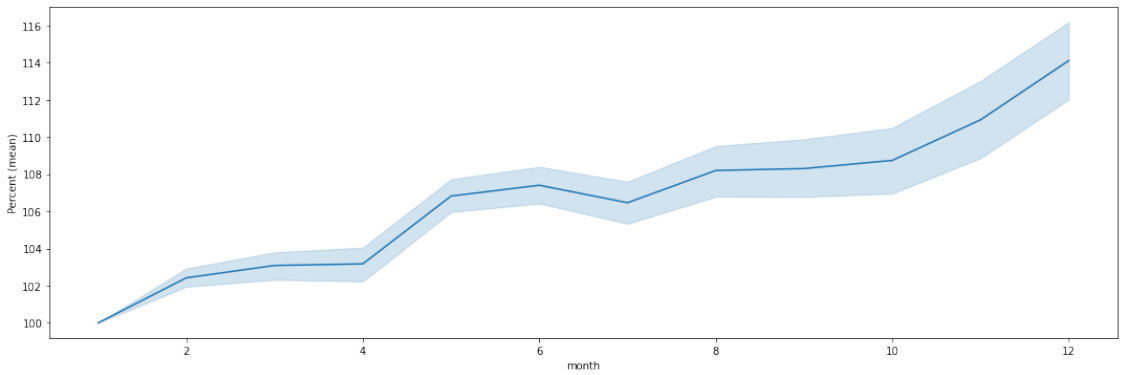
[3]:
```
plot(x=Column.MONTH, y=Column.PERCENT, data=df)
```

```
       Percent (mean)
month
1               100.0
2           102.42446
3          103.087657
4          103.174787
5          106.822324
```


Percent (mean)
month


Percent (mean)
month


Percent (mean)
month


Percent (mean)
month

1.2 Weekly stock price fluctuations within a year

[4]:
```
from analysis import get_best_week

df = get_best_week(FILENAME, YahooRange.YEARS_20, limit=LIMIT)

df
```

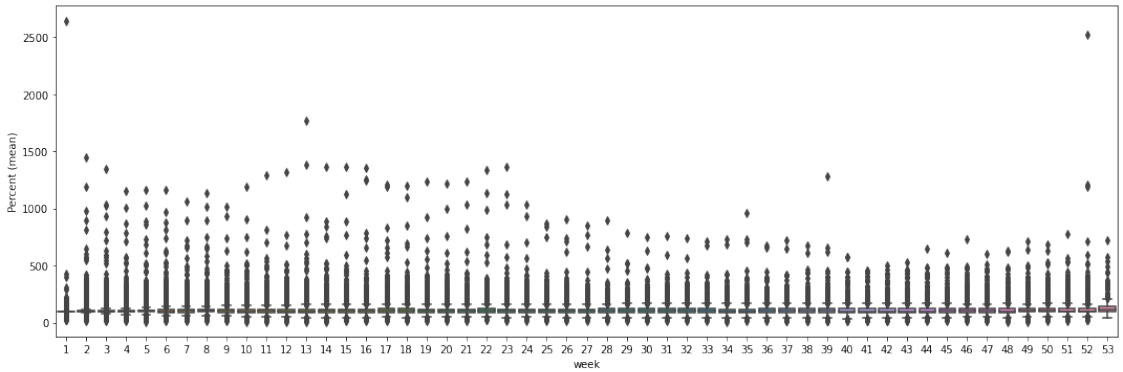
[4]:

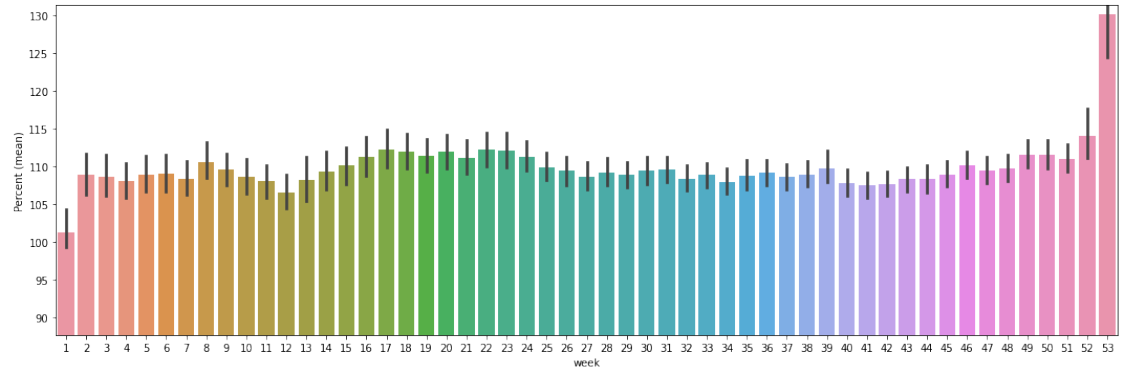
	year	week	Symbol	Percent (mean)
0	2001	1	ALV.DE	100.0
1	2001	2	ALV.DE	94.004016
2	2001	3	ALV.DE	90.689911
3	2001	4	ALV.DE	88.283991
4	2001	5	ALV.DE	90.378825
…	…	…	…	…
114280	2020	49	O2D.DE	91.401153
114281	2020	50	O2D.DE	91.401153
114282	2020	51	O2D.DE	87.216893
114283	2020	52	O2D.DE	88.291744
114284	2020	53	O2D.DE	88.483687

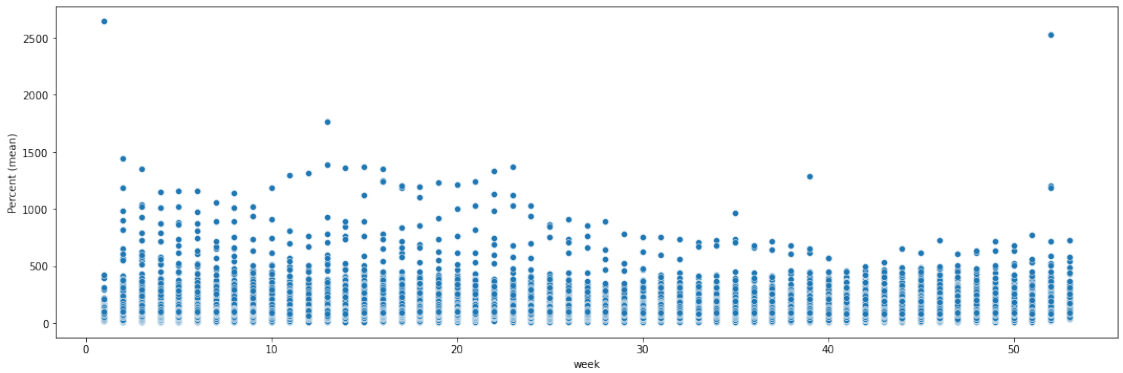
[114285 rows x 4 columns]

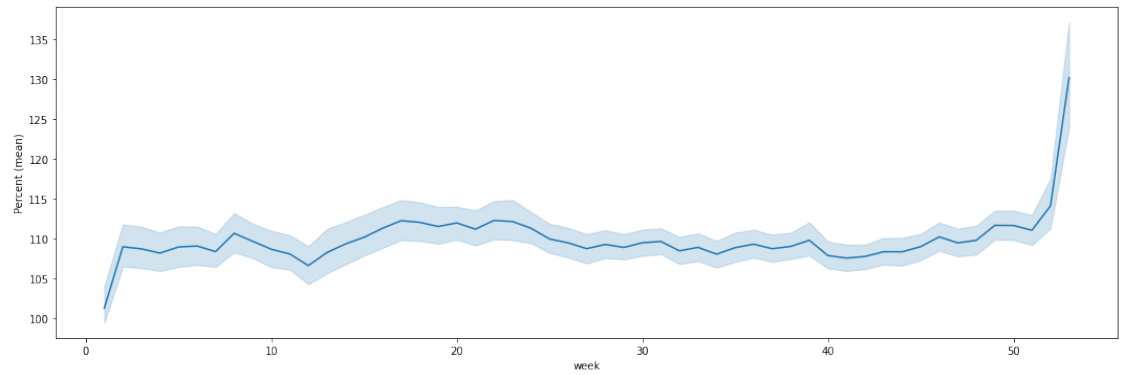
[5]:
```
plot(x=Column.WEEK, y=Column.PERCENT, data=df)
```

week	Percent (mean)
1	101.259749
2	108.959325
3	108.700405
4	108.170636
5	108.935119


Percent (mean)
week


Percent (mean)
week


Percent (mean)
week


Percent (mean)
week

1.3 Daily stock price fluctuations within a month

[6]:
```
from analysis import Column,get_best_month_day

df = get_best_month_day(FILENAME, YahooRange.YEARS_20, limit=LIMIT)

df
```

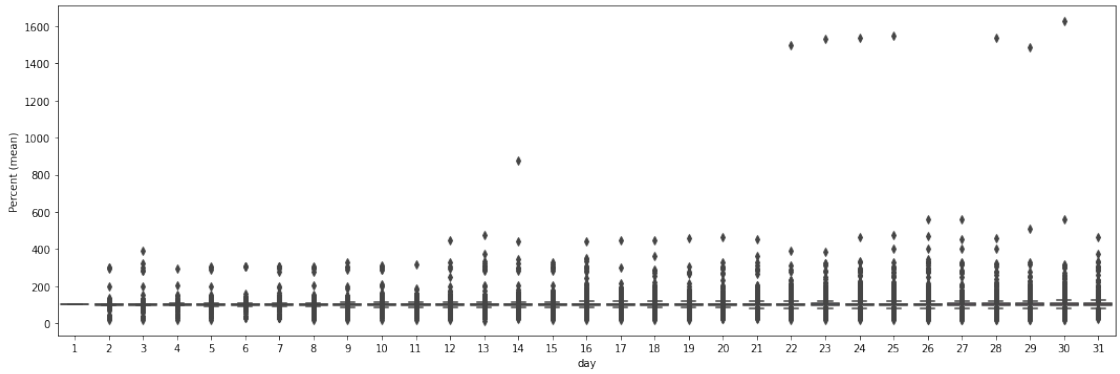
[6]:

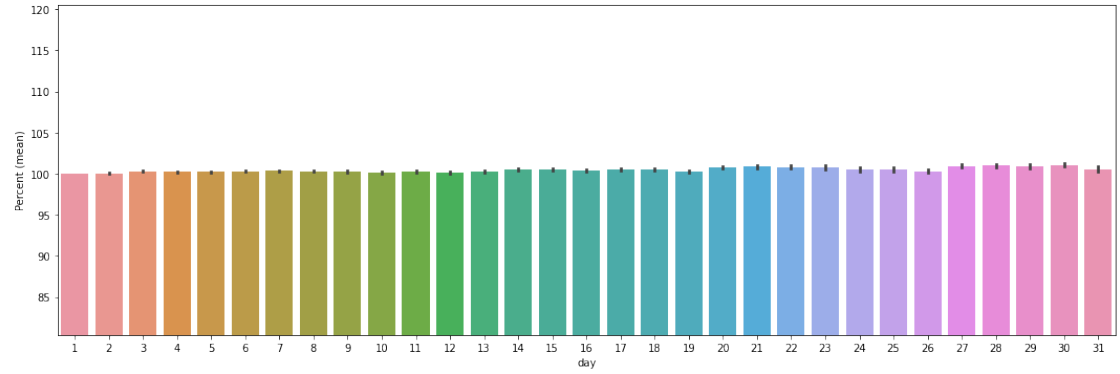
	year	month	day	Symbol	Percent (mean)
0	2007	4	3	AOX.DE	100.0
1	2007	4	4	AOX.DE	101.226999
2	2007	4	5	AOX.DE	102.453997
3	2007	4	10	AOX.DE	102.392646
4	2007	4	11	AOX.DE	101.226999
…	…	…	…	…	…
582873	2020	8	25	SBS.DE	125.450895
582874	2020	8	26	SBS.DE	125.050099
582875	2020	8	27	SBS.DE	129.058115
582876	2020	8	28	SBS.DE	119.839674
582877	2020	8	31	SBS.DE	102.204406

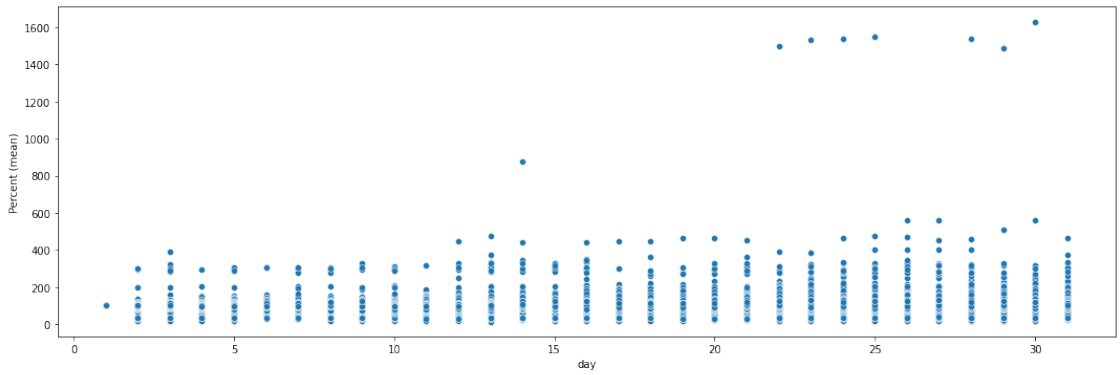
[582878 rows x 5 columns]

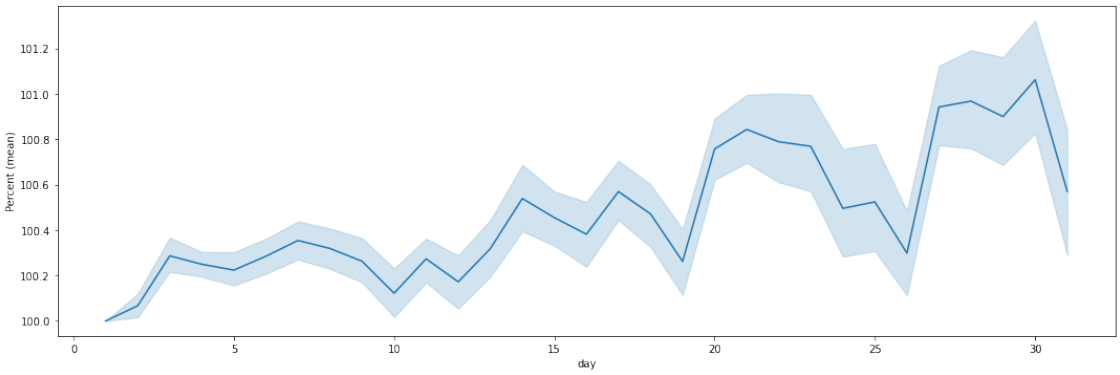
[7]:
```
plot(x=Column.DAY, y=Column.PERCENT, data=df)
```

day	Percent (mean)
1	100.0
2	100.066758
3	100.286406
4	100.249477
5	100.223694


Percent (mean)
day


Percent (mean)
day


Percent (mean)
day


Percent (mean)
day

1.4 Daily stock price fluctuations within a week

[8]:
```
from analysis import get_best_weekday


df = get_best_weekday(FILENAME, YahooRange.YEARS_20, limit=LIMIT)

df
```

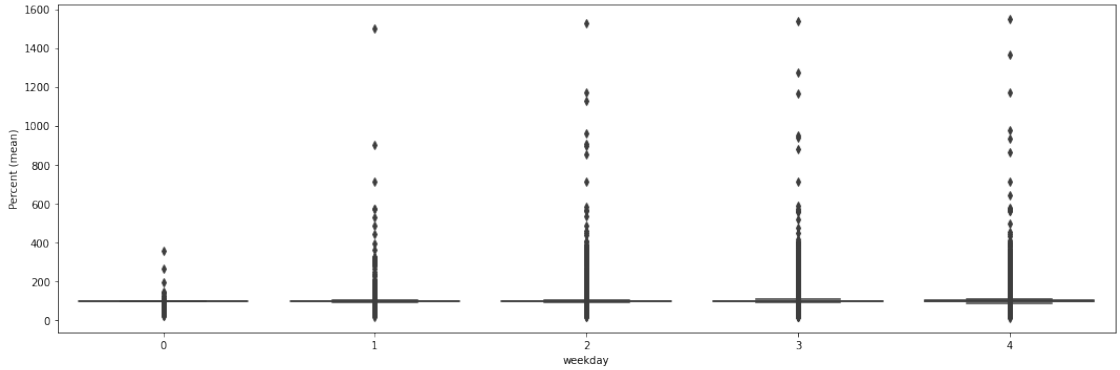
[8]:

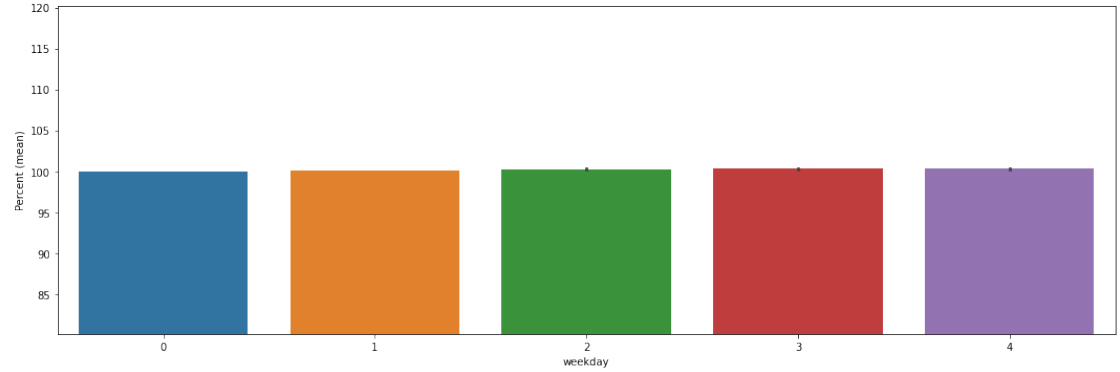
	year	week	weekday	Symbol	Percent (mean)
0	2007	14	1	AOX.DE	100.0
1	2007	14	2	AOX.DE	101.226999
2	2007	14	3	AOX.DE	102.453997
3	2007	15	1	AOX.DE	100.0
4	2007	15	2	AOX.DE	98.861591
…	…	…	…	…	…
588254	2020	42	3	SBS.DE	106.825402
588255	2020	42	4	SBS.DE	103.650798
588256	2020	53	0	SBS.DE	100.0
588257	2020	53	1	SBS.DE	100.330034
588258	2020	53	2	SBS.DE	100.0

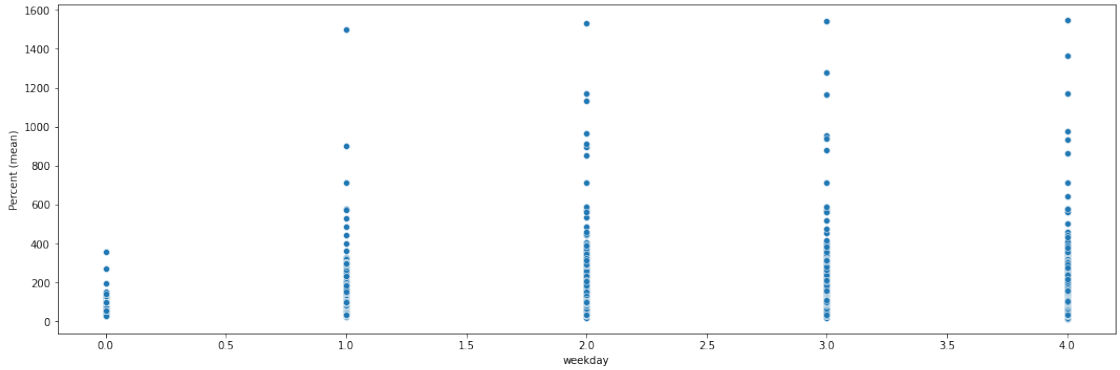
[588259 rows x 5 columns]

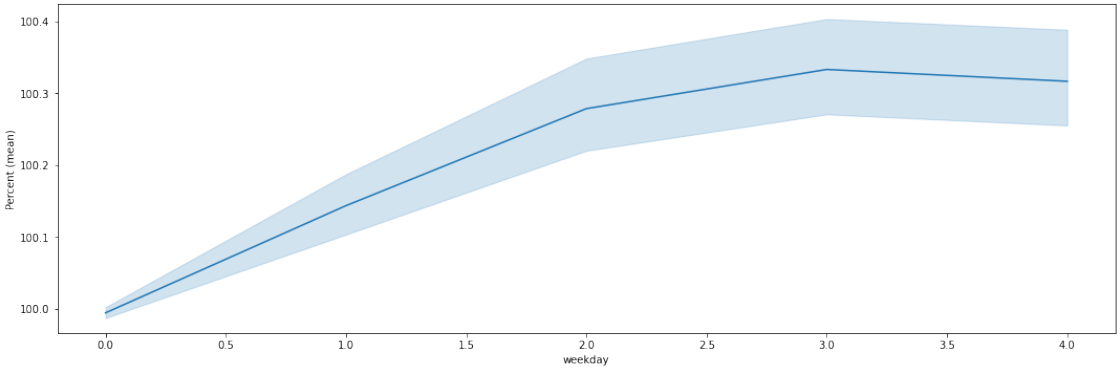
[9]:
```
plot(x=Column.WEEKDAY, y=Column.PERCENT, data=df)
```

weekday	Percent (mean)
0	99.994042
1	100.143274
2	100.278382
3	100.332932
4	100.316481


Percent (mean)
weekday


Percent (mean)
weekday


Percent (mean)
weekday


Percent (mean)
weekday

1.5 Hourly stock price fluctuations with a day

[10]:
```
from analysis import get_best_hour

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

df
```

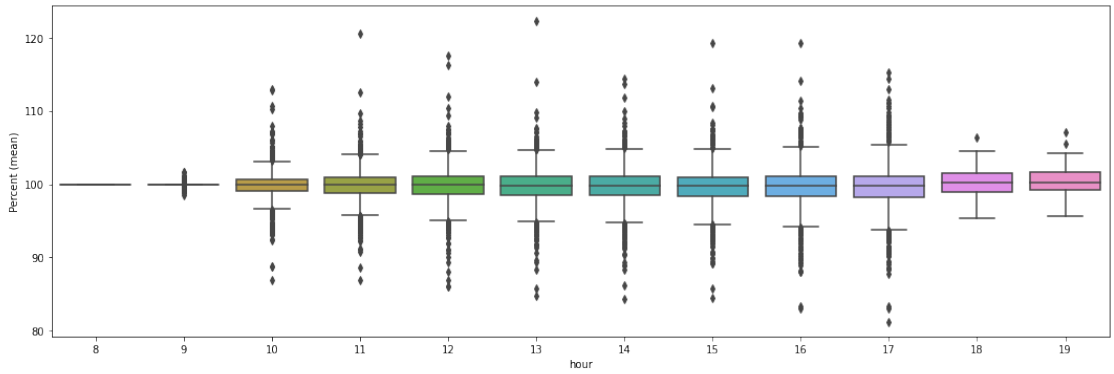
[10]:

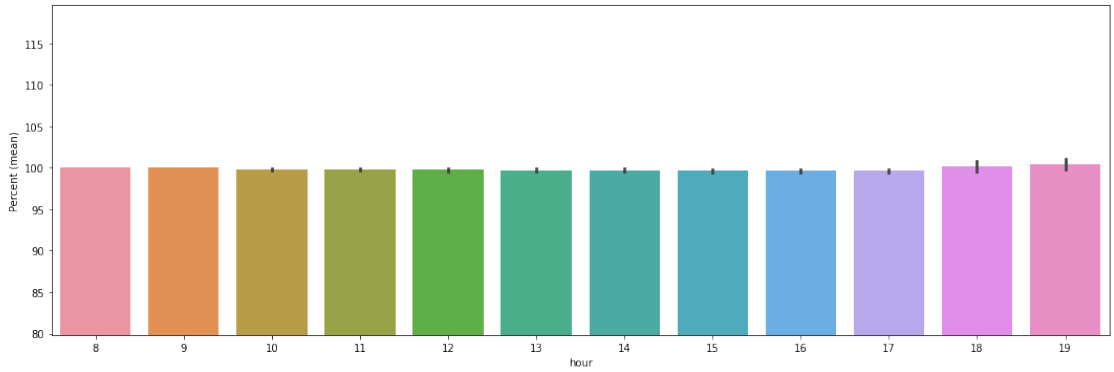
	year	week	day	hour	Symbol	Percent (mean)
0	2019	27	1	9	8TRA.DE	100.0
1	2019	27	1	10	8TRA.DE	100.883454
2	2019	27	1	11	8TRA.DE	100.469925
3	2019	27	1	12	8TRA.DE	100.977444
4	2019	27	1	13	8TRA.DE	101.165412
…	…	…	…	…	…	…
9740	2020	53	28	13	GFG.DE	107.111304
9741	2020	53	28	14	GFG.DE	106.647037
9742	2020	53	28	15	GFG.DE	106.647037
9743	2020	53	28	16	GFG.DE	107.30381
9744	2020	53	28	17	GFG.DE	107.349102

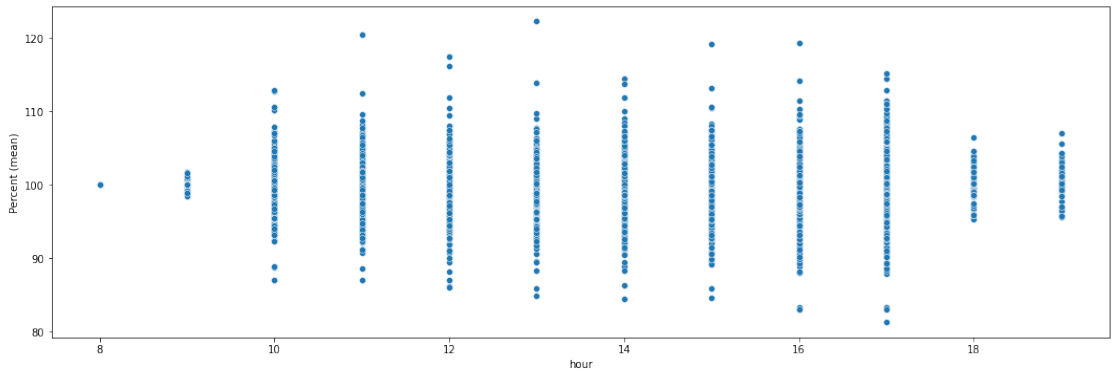
[9745 rows x 6 columns]

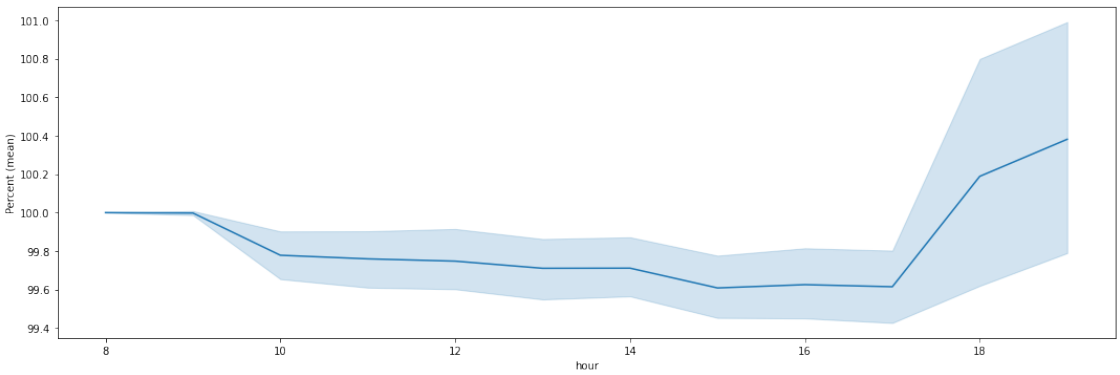
[11]:
```
plot(x=Column.HOUR, y=Column.PERCENT, data=df)
```

hour	Percent (mean)
8	100.0
9	99.997761
10	99.778236
11	99.759461
12	99.74753


Percent (mean)
hour


Percent (mean)
hour


Percent (mean)
hour


Percent (mean)
hour

1.6 Hourly and quarterly stock price fluctuations within a day

[12]:
```
from analysis import get_best_time

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

df
```

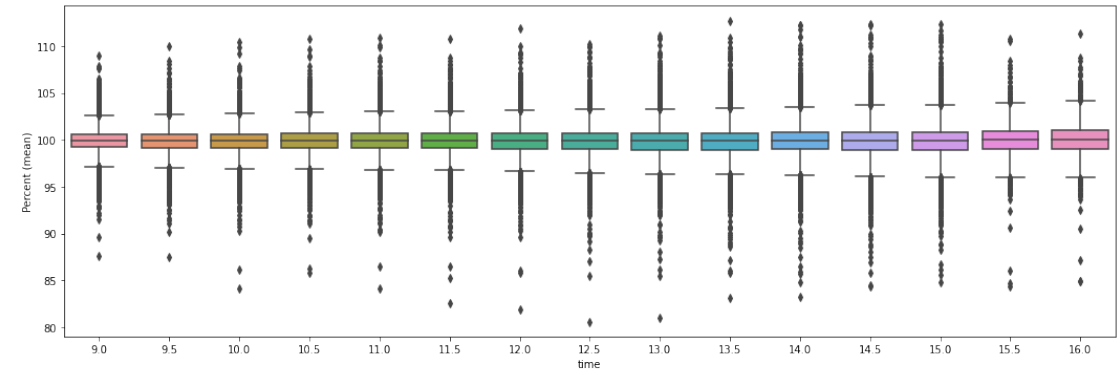
[12]:

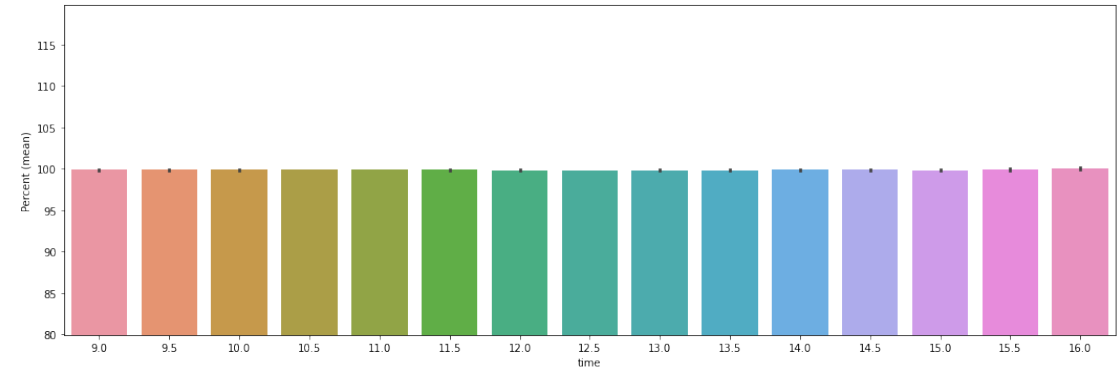
	year	week	day	hour	minute	time	Symbol	Percent (mean)
0	2021	11	18	8	0	8.0	NDX1.DE	100.0
1	2021	11	18	8	30	8.5	NDX1.DE	99.91103
2	2021	11	18	9	0	9.0	NDX1.DE	98.220642
3	2021	11	18	9	30	9.5	NDX1.DE	98.309612
4	2021	11	18	10	0	10.0	NDX1.DE	98.576514
…	…	…	…	…	…	…	…	…
107084	2021	19	11	13	0	13.0	EVK.DE	98.022788
107085	2021	19	11	13	30	13.5	EVK.DE	98.391422
107086	2021	19	11	14	0	14.0	EVK.DE	99.095173
107087	2021	19	11	14	30	14.5	EVK.DE	98.726538
107088	2021	19	11	15	0	15.0	EVK.DE	98.592493

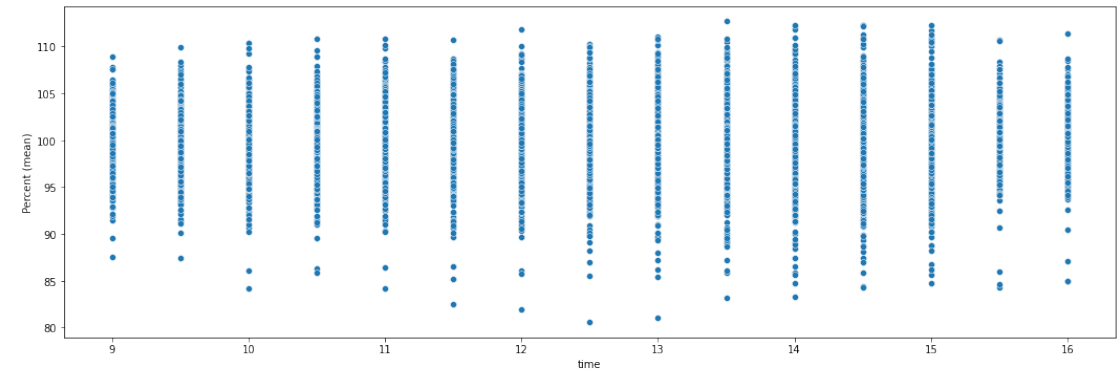
[107089 rows x 8 columns]

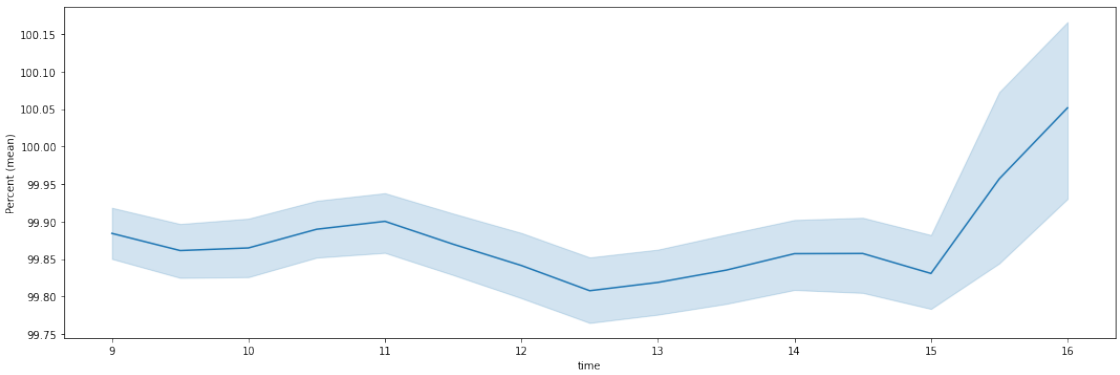
[13]:
```
# NOTE: filter extreme points, plot df first and if charts are bad try with fdf
fdf = df[df[Column.TIME].isin(np.arange(9, 16.5, 0.25))].copy()

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

time	Percent (mean)
9.0	99.88415
9.5	99.861226
10.0	99.864699
10.5	99.889672
11.0	99.900217


Percent (mean)
time
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
15.5
16.0


Percent (mean)
time


Percent (mean)
time


Percent (mean)
time

1.7 Quarterly stock price fluctuations within an hour

[14]:
```
from analysis import get_best_quarter

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

df
```

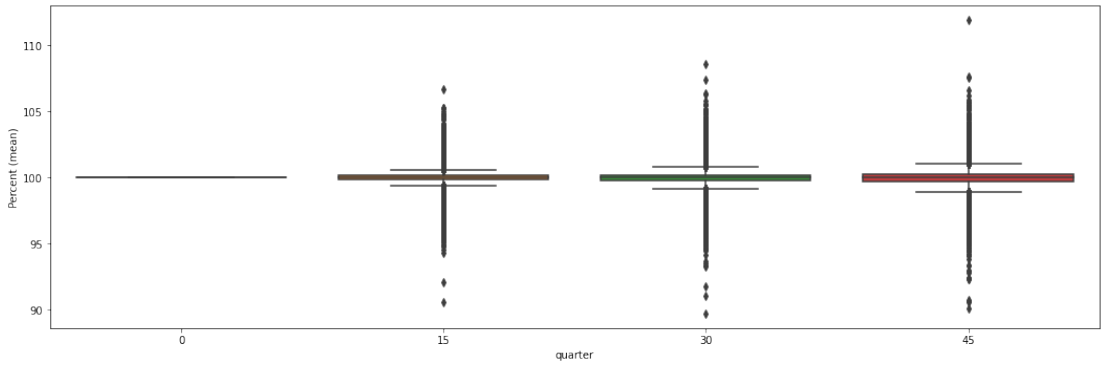
[14]:

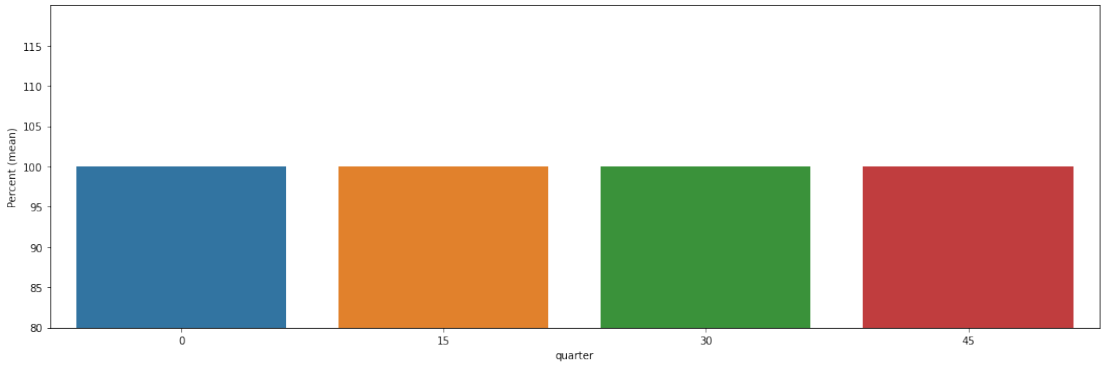
	year	week	day	hour	minute	quarter	Symbol	Percent (mean)
0	2021	11	18	8	0	0	NDX1.DE	100.0
1	2021	11	18	8	15	15	NDX1.DE	100.0
2	2021	11	18	8	30	30	NDX1.DE	99.91103
3	2021	11	18	8	45	45	NDX1.DE	99.110325
4	2021	11	18	9	0	0	NDX1.DE	100.0
…	…	…	…	…	…	…	…	…
190160	2021	19	11	15	15	15	SBS.DE	100.372441
190161	2021	19	11	7	0	0	SBS.DE	100.0
190162	2021	19	11	7	15	15	SBS.DE	98.007242
190163	2021	19	11	7	30	30	SBS.DE	98.007242
190164	2021	19	11	7	45	45	SBS.DE	98.007242

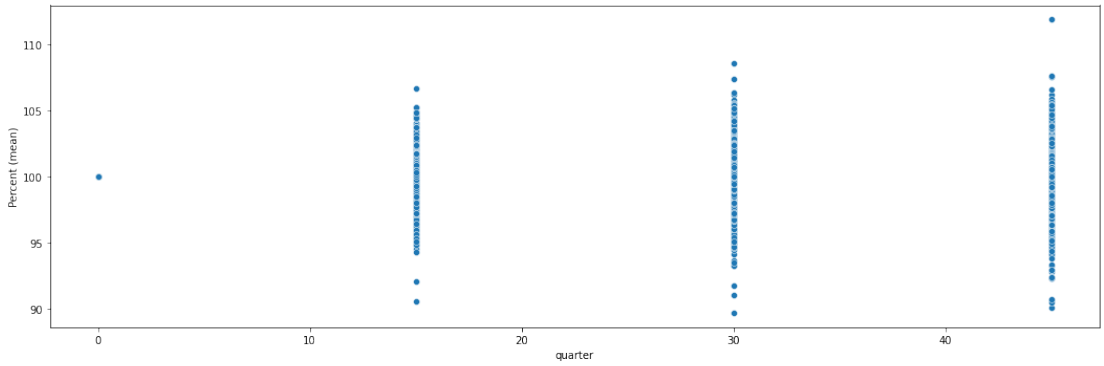
[190165 rows x 8 columns]

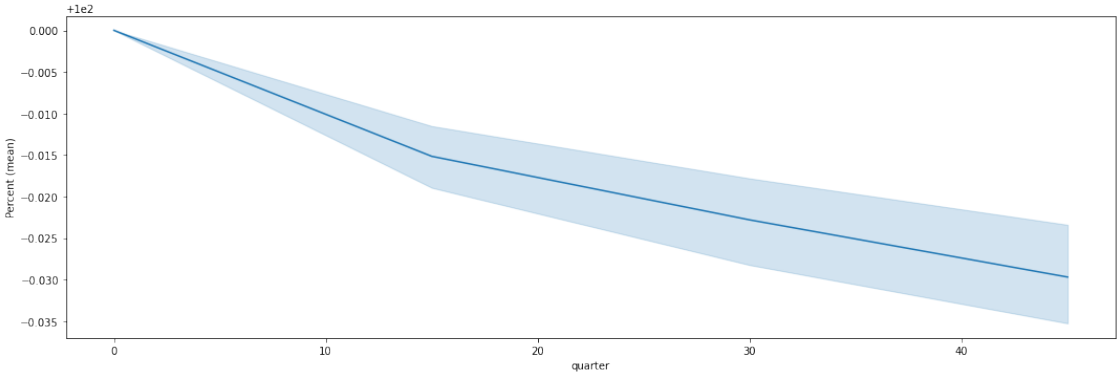
[15]:
```
plot(x=Column.QUARTER, y=Column.PERCENT, data=df)
```

quarter	Percent (mean)
0	100.0
15	99.984836
30	99.977188
45	99.97031


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


Percent (mean)
115
110
105
100
95
90
85
80
0
15
30
45
quarter


Percent (mean)
110
105
100
95
90
0
10
20
30
40
quarter


+1e2
Percent (mean)
0.000
−0.005
−0.010
−0.015
−0.020
−0.025
−0.030
−0.035
0
10
20
30
40
quarter

[ ]: