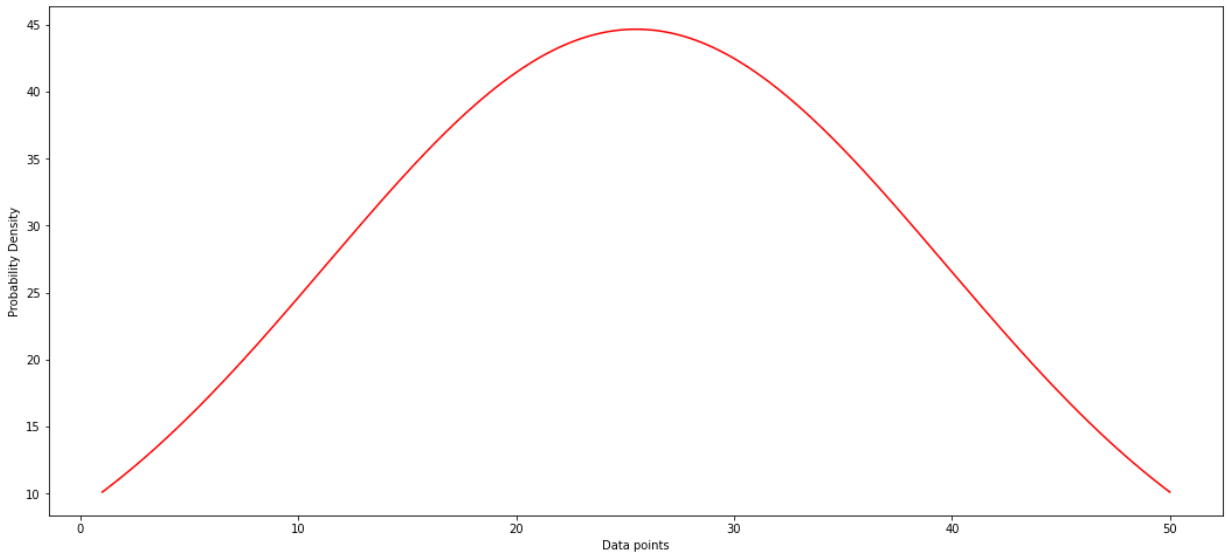
Нормальное распределение

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import pandas as pd
plt.rcParams['figure.figsize'] = 18, 8
import seaborn as sns
import scipy.stats as stats
import random
import pylab
```

In [2]:
```
# Creating a series of data of in range of 1-50.
x = np.linspace(1,50,200)

#Creating a Function.
def normal_dist(x , mean , sd):
    prob_density = (np.pi*sd) * np.exp(-0.5*((x-mean)/sd)**2)
    return prob_density

#Calculate mean and Standard deviation.
mean = np.mean(x)
sd = np.std(x)

#Apply function to the data.
pdf = normal_dist(x,mean,sd)

#Plotting the Results
plt.plot(x,pdf , color = 'red')
plt.xlabel('Data points')
plt.ylabel('Probability Density')
```

Out[2]:
```
Text(0, 0.5, 'Probability Density')
```



In [3]:
```
x
```

Out[3]:
```
array([ 1.        ,  1.24623116,  1.49246231,  1.73869347,  1.98492462,
        2.23115578,  2.47738693,  2.72361809,  2.96984925,  3.2160804 ,
        3.46231156,  3.70854271,  3.95477387,  4.20100503,  4.44723618,
        4.69346734,  4.93969849,  5.18592965,  5.4321608 ,  5.67839196,
```

```
 5.92462312,  6.17085427,  6.41708543,  6.66331658,  6.90954774,
 7.15577889,  7.40201005,  7.64824121,  7.89447236,  8.14070352,
 8.38693467,  8.63316583,  8.87939698,  9.12562814,  9.3718593 ,
 9.61809045,  9.86432161, 10.11055276, 10.35678392, 10.60301508,
10.84924623, 11.09547739, 11.34170854, 11.5879397 , 11.83417085,
12.08040201, 12.32663317, 12.57286432, 12.81909548, 13.06532663,
13.31155779, 13.55778894, 13.8040201 , 14.05025126, 14.29648241,
14.54271357, 14.78894472, 15.03517588, 15.28140704, 15.52763819,
15.77386935, 16.0201005 , 16.26633166, 16.51256281, 16.75879397,
17.00502513, 17.25125628, 17.49748744, 17.74371859, 17.98994975,
18.2361809 , 18.48241206, 18.72864322, 18.97487437, 19.22110553,
19.46733668, 19.71356784, 19.95979899, 20.20603015, 20.45226131,
20.69849246, 20.94472362, 21.19095477, 21.43718593, 21.68341709,
21.92964824, 22.1758794 , 22.42211055, 22.66834171, 22.91457286,
23.16080402, 23.40703518, 23.65326633, 23.89949749, 24.14572864,
24.3919598 , 24.63819095, 24.88442211, 25.13065327, 25.37688442,
25.62311558, 25.86934673, 26.11557789, 26.36180905, 26.6080402 ,
26.85427136, 27.10050251, 27.34673367, 27.59296482, 27.83919598,
28.08542714, 28.33165829, 28.57788945, 28.8241206 , 29.07035176,
29.31658291, 29.56281407, 29.80904523, 30.05527638, 30.30150754,
30.54773869, 30.79396985, 31.04020101, 31.28643216, 31.53266332,
31.77889447, 32.02512563, 32.27135678, 32.51758794, 32.7638191 ,
33.01005025, 33.25628141, 33.50251256, 33.74874372, 33.99497487,
34.24120603, 34.48743719, 34.73366834, 34.9798995 , 35.22613065,
35.47236181, 35.71859296, 35.96482412, 36.21105528, 36.45728643,
36.70351759, 36.94974874, 37.1959799 , 37.44221106, 37.68844221,
37.93467337, 38.18090452, 38.42713568, 38.67336683, 38.91959799,
39.16582915, 39.4120603 , 39.65829146, 39.90452261, 40.15075377,
40.39698492, 40.64321608, 40.88944724, 41.13567839, 41.38190955,
41.6281407 , 41.87437186, 42.12060302, 42.36683417, 42.61306533,
42.85929648, 43.10552764, 43.35175879, 43.59798995, 43.84422111,
44.09045226, 44.33668342, 44.58291457, 44.82914573, 45.07537688,
45.32160804, 45.5678392 , 45.81407035, 46.06030151, 46.30653266,
46.55276382, 46.79899497, 47.04522613, 47.29145729, 47.53768844,
47.7839196 , 48.03015075, 48.27638191, 48.52261307, 48.76884422,
49.01507538, 49.26130653, 49.50753769, 49.75376884, 50.        ])
```

Представление через pandas

In [4]:

```
df = pd.DataFrame()
df['x'] = x
df
```

Out[4]:

	x
0	1.000000
1	1.246231
2	1.492462
3	1.738693
4	1.984925
...	...
195	49.015075
196	49.261307
197	49.507538
198	49.753769

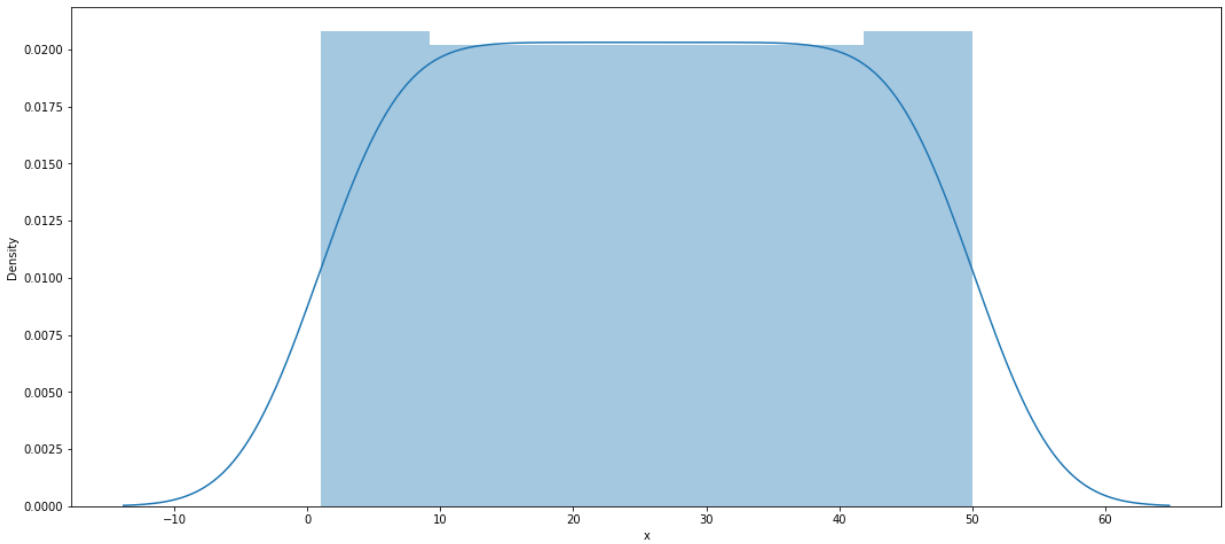
	x
199	50.000000

200 rows × 1 columns

Представление данных с помощью гистограмм

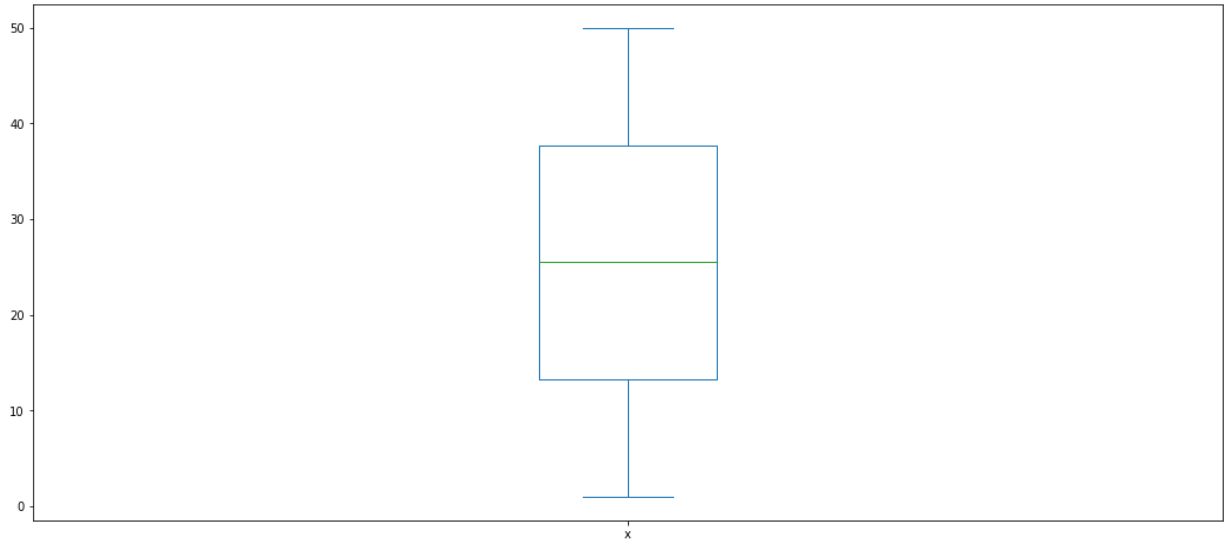
In [5]:
```
sns_plot = sns.distplot(df.x, label='начальные данные')
fig = sns_plot.get_figure()
```

```
/Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/
seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated funct
ion and will be removed in a future version. Please adapt your code to use eit
her `displot` (a figure-level function with similar flexibility) or `histplot`
(an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



BoxPlot или Ящик с усами

In [6]:
```
_, bp = df.x.plot.box(return_type='both')
outliers = [flier.get_ydata() for flier in bp["fliers"]][0]
```



In [7]:
```
outliers
```

Out[7]:
```
array([], dtype=float64)
```

Аномалии

In [8]:
```
df.x[199] = 100
df.x[198] = 88
```

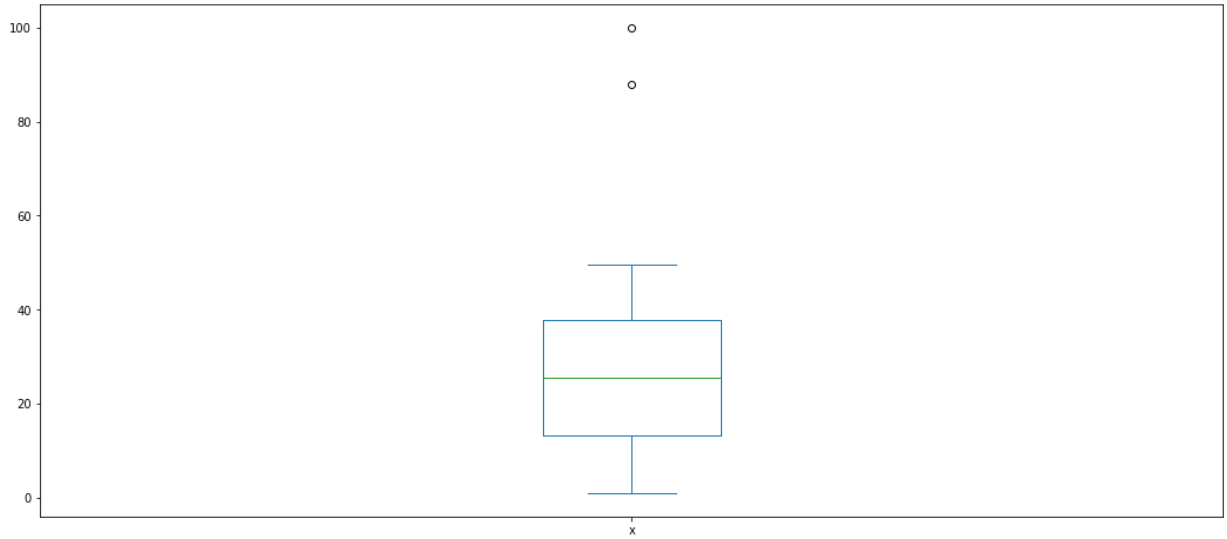
In [9]:
```
df
```

Out[9]:

	x
0	1.000000
1	1.246231
2	1.492462
3	1.738693
4	1.984925
...	...
195	49.015075
196	49.261307
197	49.507538
198	88.000000
199	100.000000

200 rows × 1 columns

In [10]:
```
_, bp = df.x.plot.box(return_type='both')
outliers = [flier.get_ydata() for flier in bp["fliers"]][0]
```



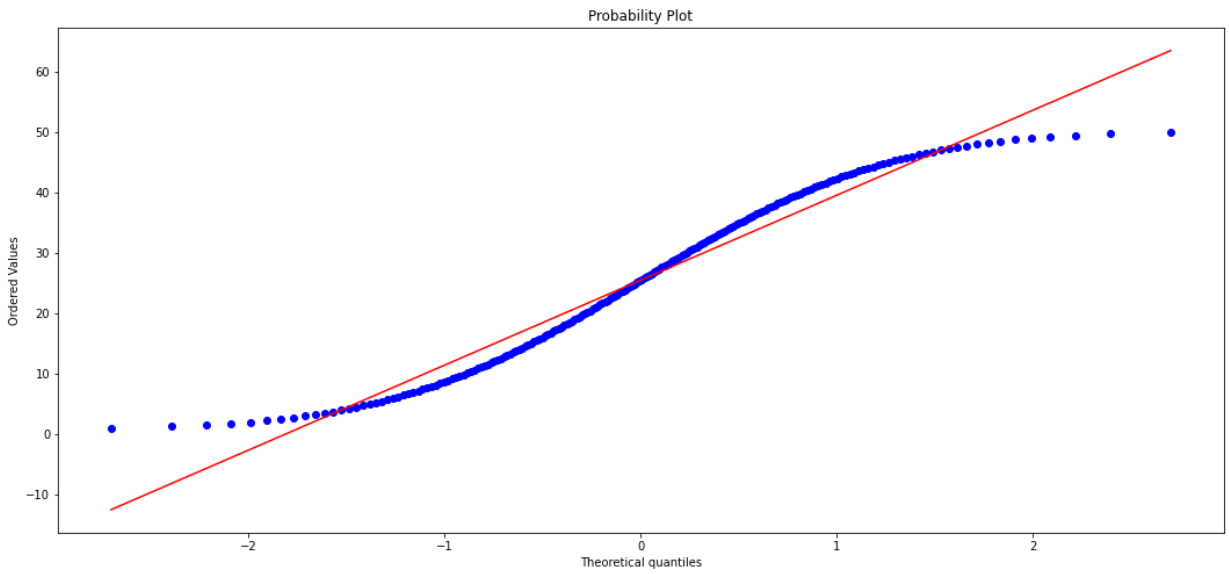
In [11]:
```
outliers
```

Out[11]:
```
array([ 88., 100.])
```

In [12]:
```
x = np.linspace(1,50,200)
df = pd.DataFrame()
df['x'] = x
```

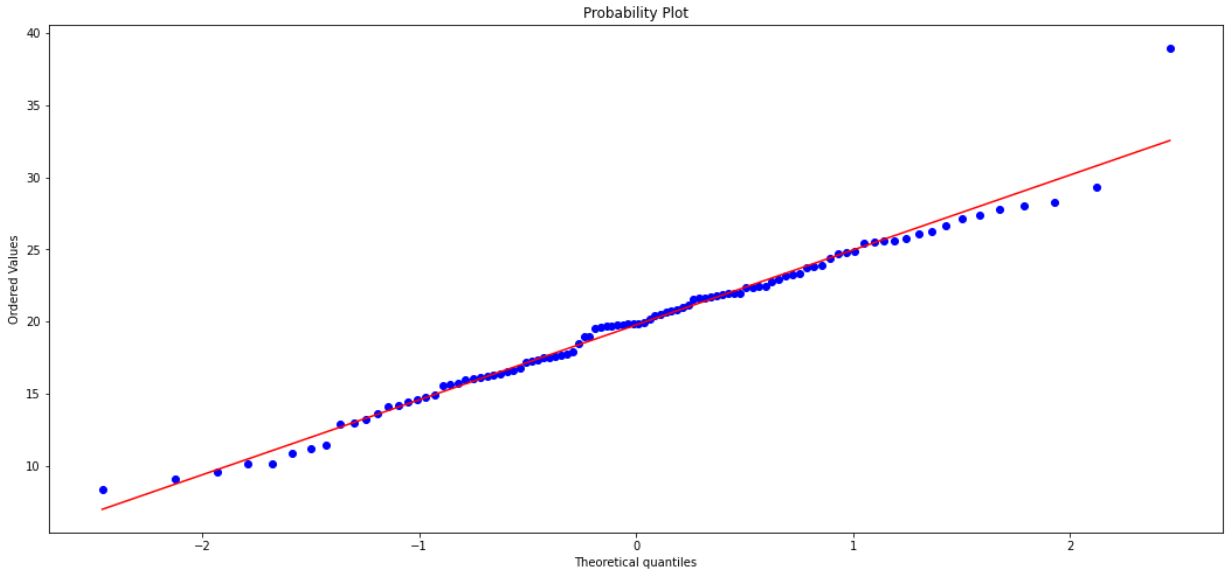
Квантиль-квантиль график

In [13]:
```
stats.probplot(df.x, dist="norm", plot=pylab)
pylab.show()
```



In [14]:
```
measurements = np.random.normal(loc = 20, scale = 5, size=100)
```

In [15]:
```
stats.probplot(measurements, dist="norm", plot=pylab)
pylab.show()
```



Валютные рынки

Индекс S&P500 https://www.finam.ru/profile/mirovye-indeksy/sandp-500/export/ Золото https://www.finam.ru/profile/tovary/gold/export/ Биткоин/USD https://www.finam.ru/quote/cryptocurrencies/bch-usd/ Нефть.Брент https://www.finam.ru/profile/tovary/brent/export/ Швецария франк/ USD https://www.finam.ru/profile/forex/chf-usd/export/ Euro/USD https://www.finam.ru/profile/forex/eur-usd/export/ Платина https://www.finam.ru/profile/tovary/platinum/export/ Норвегия крона/USD https://www.finam.ru/profile/forex/usd-nok/export/

In [17]:
```
nz = pd.read_csv("GBPUSD.csv",sep = ';')
pred1 = pd.read_csv("1_CHFUSD.csv", sep = ';')
pred2 = pd.read_csv("2_comex.GC.csv", sep = ';')
pred3 = pd.read_csv("3_EURUSD.csv", sep = ';')
pred4 = pd.read_csv("4_GDAX.BCH-USD.csv",sep = ';')
pred5 = pd.read_csv("5_ICE.BRN.csv",sep = ';')
pred6 = pd.read_csv("6_NYMEX.PL.csv",sep = ';')
pred7 = pd.read_csv("7_SANDP-500.csv",sep = ';')
pred8 = pd.read_csv("8_USDNOK.csv",sep = ';')
```

In [18]:
```
nz = pd.read_csv("GBPUSD.csv",sep = ';')
```

In [19]:
```
nz
```

Out[19]:

	<DATE>	<CLOSE.GBPUSD>
0	25.05.2011	1.6278
1	26.05.2011	1.6389
2	27.05.2011	1.6478
3	28.05.2011	1.6506
4	29.05.2011	1.6500
...	...	...

	<DATE>	<CLOSE.GBPUSD>
3523	21.05.2021	1.4147
3524	23.05.2021	1.4139
3525	24.05.2021	1.4155
3526	25.05.2021	1.4142
3527	26.05.2021	1.4114

3528 rows × 2 columns

In [20]:
```
pred1
```

Out[20]:

	<DATE>	<CLOSE.1_CHFUSD>
0	10.09.2013	1.0698
1	11.09.2013	1.0751
2	12.09.2013	1.0743
3	13.09.2013	1.0758
4	14.09.2013	1.0748
...	...	...
2705	21.05.2021	1.1134
2706	23.05.2021	1.1138
2707	24.05.2021	1.1143
2708	25.05.2021	1.1163
2709	26.05.2021	1.1132

2710 rows × 2 columns

In [21]:
```
M = nz.merge(pred1, left_on='<DATE>', right_on='<DATE>')
```

In [22]:
```
M = M.merge(pred2, left_on='<DATE>', right_on='<DATE>')
M = M.merge(pred3, left_on='<DATE>', right_on='<DATE>')
M = M.merge(pred4, left_on='<DATE>', right_on='<DATE>')
M = M.merge(pred5, left_on='<DATE>', right_on='<DATE>')
M = M.merge(pred6, left_on='<DATE>', right_on='<DATE>')
M = M.merge(pred7, left_on='<DATE>', right_on='<DATE>')
M = M.merge(pred8, left_on='<DATE>', right_on='<DATE>')
```

In [23]:
```
M
```

Out[23]:

	<DATE>	<CLOSE.GBPUSD>	<CLOSE.1_CHFUSD>	<CLOSE.2_comex.GC>	<CLOSE.3_EU
0	20.12.2017	1.3386	1.01350	1269.6	
1	21.12.2017	1.3386	1.01220	1270.5	
2	22.12.2017	1.3364	1.01080	1279.1	

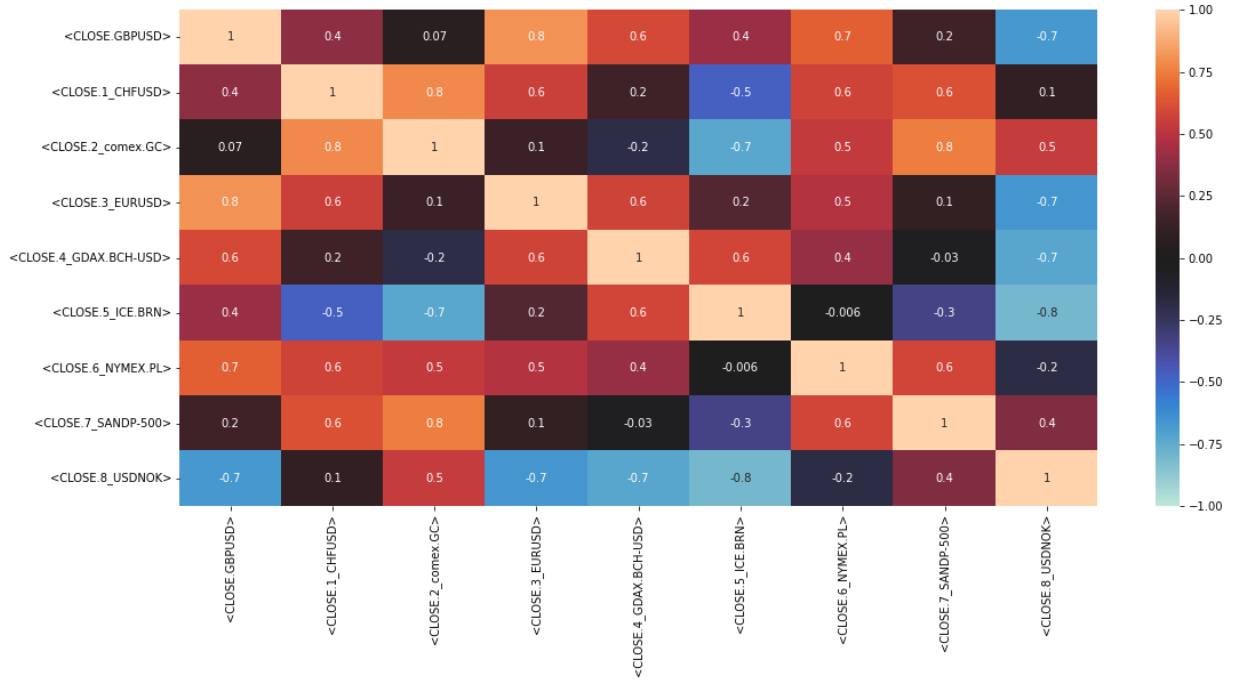
	<DATE>	<CLOSE.GBPUSD>	<CLOSE.1_CHFUSD>	<CLOSE.2_comex.GC>	<CLOSE.3_EU
3	26.12.2017	1.3378	1.01050	1287.1	
4	27.12.2017	1.3402	1.01330	1295.6	
...	...	...	...	...	
836	20.05.2021	1.4184	1.11394	1874.5	
837	21.05.2021	1.4147	1.11340	1881.8	
838	24.05.2021	1.4155	1.11430	1878.1	
839	25.05.2021	1.4142	1.11630	1898.9	
840	26.05.2021	1.4114	1.11320	1896.1	

841 rows × 10 columns

spearman

In [24]:
```
sns.heatmap(M.corr(method ='spearman'), annot = True, fmt='.1g',vmin=-1, vmax
```

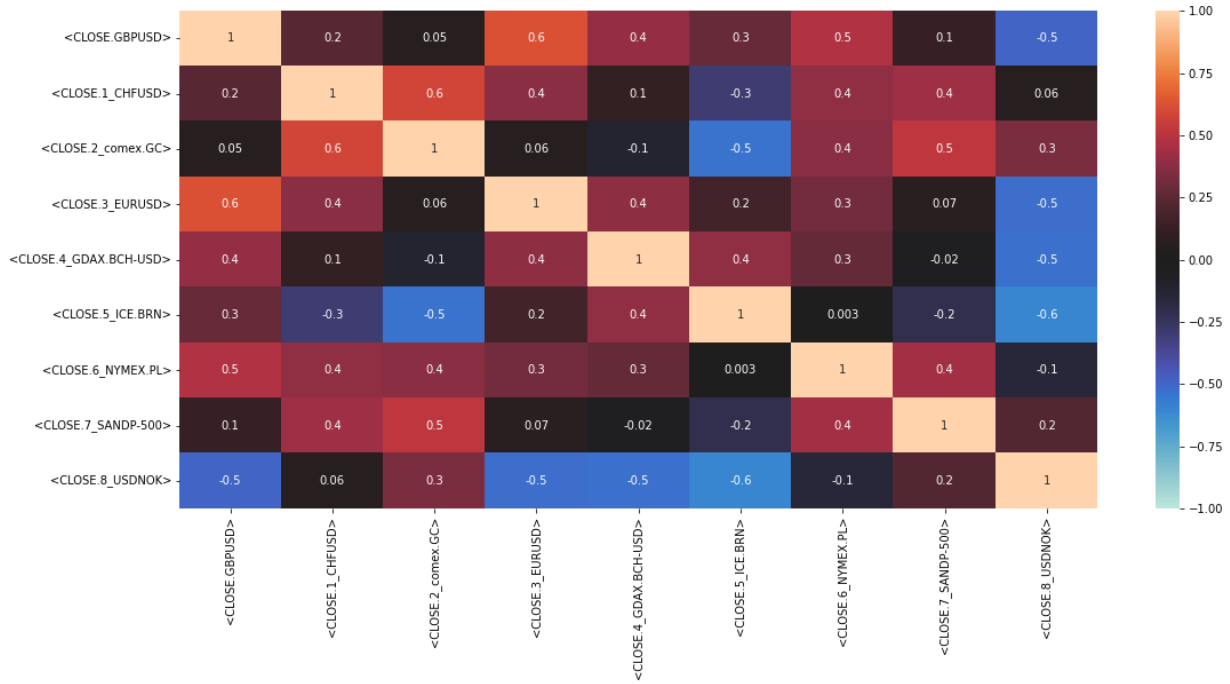
Out[24]: <AxesSubplot:>



kendall

In [25]:
```
sns.heatmap(M.corr(method ='kendall'), annot = True, fmt='.1g',vmin=-1, vmax=
```

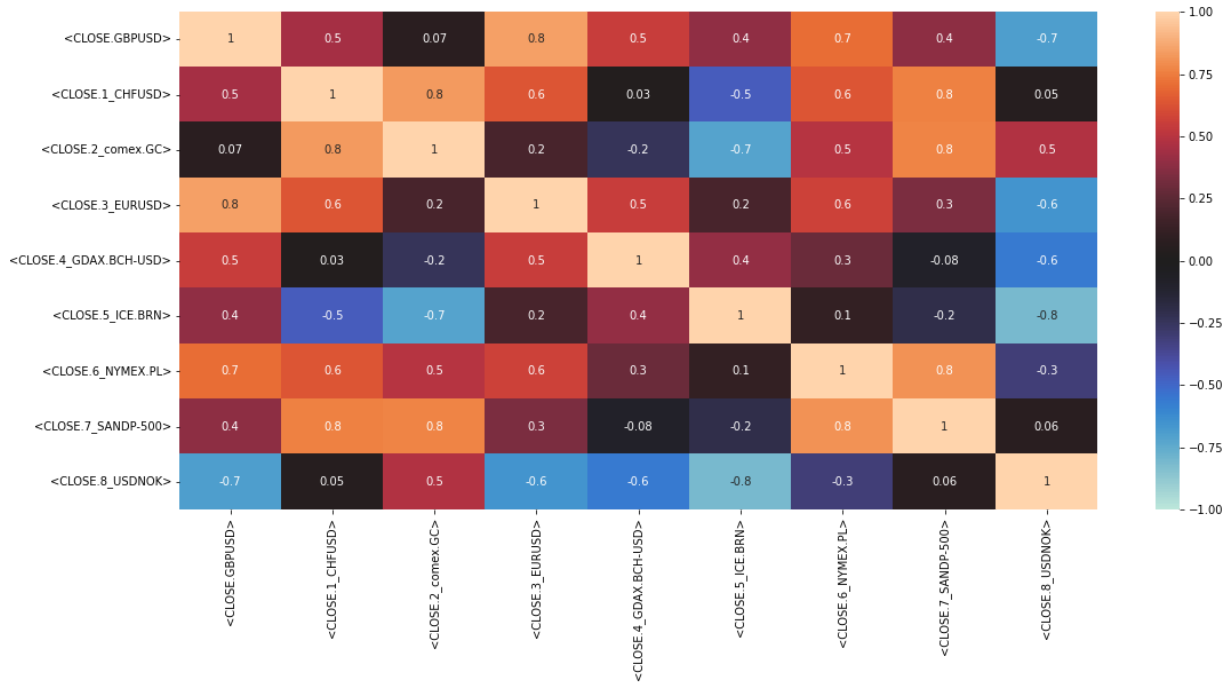
Out[25]: <AxesSubplot:>



pearson

In [26]:
```
sns.heatmap(M.corr(method ='pearson'), annot = True, fmt='.1g',vmin=-1, vmax=
```

Out[26]: <AxesSubplot:>



In [ ]: