Assignment

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
ass = pd.read_csv("assignment_data.csv")
ass.head()
```

Out[ ]:

	Domain Code	Domain	Area Code	Area	Element Code	Element	Item Code	Item	Year Code	Year	Unit	Value	Flag	Flag Description	Note
0	FS	Suite of Food Security Indicators	351	China	6127	Value	21030	Per capita food production variability (consta...	2000	2000	1000 I$	NaN	O	Missing value	NaN
1	FS	Suite of Food Security Indicators	351	China	6127	Value	21030	Per capita food production variability (consta...	2001	2001	1000 I$	8.0	E	Estimated value	NaN
2	FS	Suite of Food Security Indicators	351	China	6127	Value	21030	Per capita food production variability (consta...	2002	2002	1000 I$	5.8	E	Estimated value	NaN
3	FS	Suite of Food Security Indicators	351	China	6127	Value	21030	Per capita food production variability (consta...	2003	2003	1000 I$	3.5	E	Estimated value	NaN
4	FS	Suite of Food Security Indicators	351	China	6127	Value	21030	Per capita food production variability (consta...	2004	2004	1000 I$	6.5	E	Estimated value	NaN

In [ ]:
```
ass.describe()
```

Out[ ]:

	Area Code	Element Code	Item Code	Year Code	Year	Value	Note
count	22.0	22.0	22.0	22.000000	22.000000	19.000000	0.0
mean	351.0	6127.0	21030.0	2010.500000	2010.500000	4.573684	NaN
std	0.0	0.0	0.0	6.493587	6.493587	1.504010	NaN
min	351.0	6127.0	21030.0	2000.000000	2000.000000	2.400000	NaN
25%	351.0	6127.0	21030.0	2005.250000	2005.250000	3.550000	NaN
50%	351.0	6127.0	21030.0	2010.500000	2010.500000	4.600000	NaN
75%	351.0	6127.0	21030.0	2015.750000	2015.750000	5.450000	NaN
max	351.0	6127.0	21030.0	2021.000000	2021.000000	8.000000	NaN

In [ ]:
```
new_ass = ass.drop(['Domain Code','Area Code'], axis=1)            # dropping few columns to make a new dataset
new_ass.head()
```

Out[ ]:

	Domain	Area	Element Code	Element	Item Code	Item	Year Code	Year	Unit	Value	Flag	Flag Description	Note
0	Suite of Food Security Indicators	China	6127	Value	21030	Per capita food production variability (consta...	2000	2000	1000 I$	NaN	O	Missing value	NaN
1	Suite of Food Security Indicators	China	6127	Value	21030	Per capita food production variability (consta...	2001	2001	1000 I$	8.0	E	Estimated value	NaN
2	Suite of Food Security Indicators	China	6127	Value	21030	Per capita food production variability (consta...	2002	2002	1000 I$	5.8	E	Estimated value	NaN
3	Suite of Food Security Indicators	China	6127	Value	21030	Per capita food production variability (consta...	2003	2003	1000 I$	3.5	E	Estimated value	NaN
4	Suite of Food Security Indicators	China	6127	Value	21030	Per capita food production variability (consta...	2004	2004	1000 I$	6.5	E	Estimated value	NaN

In [ ]:
```
ass.mean()
```

```
C:\Users\Me\AppData\Local\Temp\ipykernel_7908\1483967947.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this w
ill raise TypeError.  Select only valid columns before calling the reduction.
  ass.mean()
```

Out[ ]:
```
Area Code         351.000000
Element Code     6127.000000
Item Code       21030.000000
Year Code        2010.500000
Year             2010.500000
Value               4.573684
Note                     NaN
dtype: float64
```

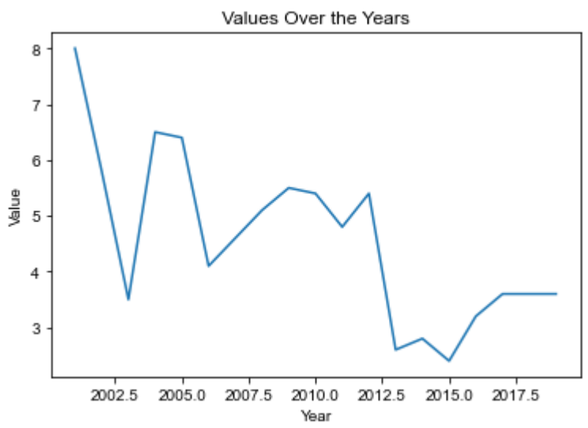
In [ ]:
```
ass.head(2)
```

Out[ ]:

	Domain Code	Domain	Area Code	Area	Element Code	Element	Item Code	Item	Year Code	Year	Unit	Value	Flag	Flag Description	Note
0	FS	Suite of Food Security Indicators	351	China	6127	Value	21030	Per capita food production variability (consta...	2000	2000	1000 I$	NaN	O	Missing value	NaN
1	FS	Suite of Food Security Indicators	351	China	6127	Value	21030	Per capita food production variability (consta...	2001	2001	1000 I$	8.0	E	Estimated value	NaN

> **Line Plots**

In [ ]:
```
ass = pd.read_csv("assignment_data.csv")
sns.lineplot(x="Year", y="Value", data=ass)
plt.title("Values Over the Years")
sns.set_style("darkgrid")
plt.figure(figsize=(8,6))
plt.show()
```




```
<Figure size 576x432 with 0 Axes>
```

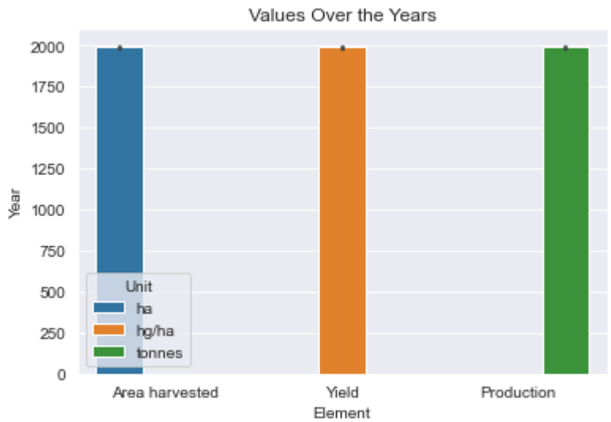
In [ ]:
```
ass2 = pd.read_csv("ass_data2.csv")
ass2.head(2)
```

Out[ ]:

	Domain Code	Domain	Area Code	Area	Element Code	Element	Item Code	Item	Year Code	Year	Unit	Value	Flag	Flag Description
0	QCL	Crops and livestock products	106	Italy	5312	Area harvested	1717	Cereals, Total	1961	1961	ha	6387203	E	Estimated value
1	QCL	Crops and livestock products	106	Italy	5419	Yield	1717	Cereals, Total	1961	1961	hg/ha	21815	E	Estimated value

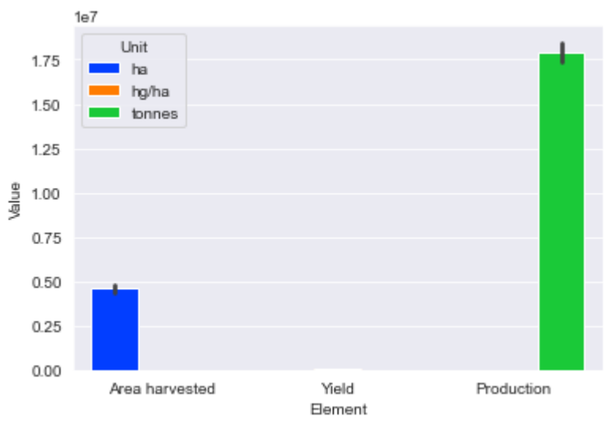
In [ ]:
```
ass2 = pd.read_csv("ass_data2.csv")

# draw a bar plot
sns.barplot(x="Element", y="Year", hue="Unit", data=ass2)
plt.title("Values Over the Years")
plt.show()
```



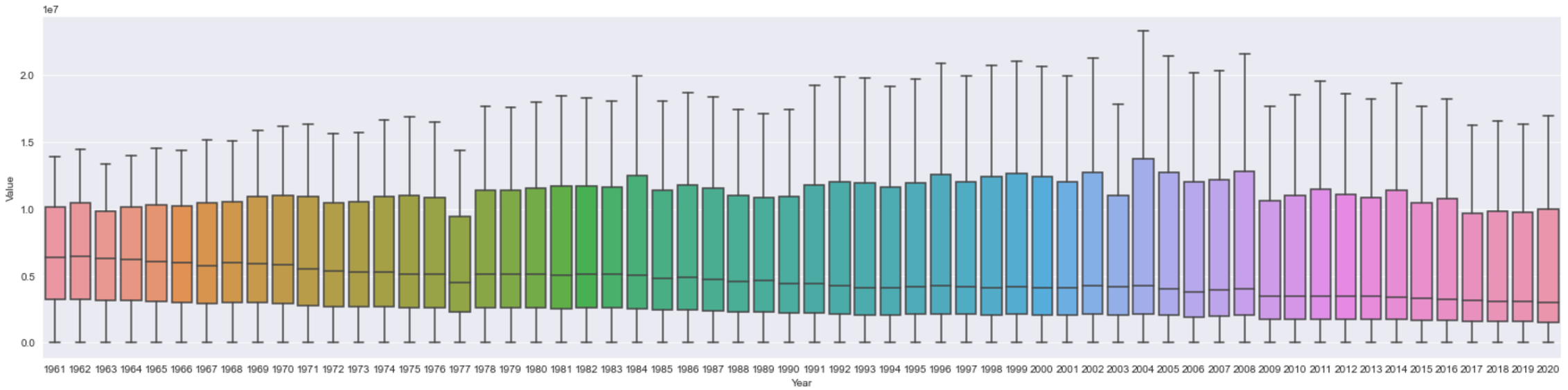
In [ ]:
```
import numpy as np
# draw a bar plot
sns.barplot(x="Element", y="Value", hue="Unit", data=ass2, color="red",
            palette="bright", saturation=5.2)
```

Out[ ]:
```
<AxesSubplot:xlabel='Element', ylabel='Value'>
```



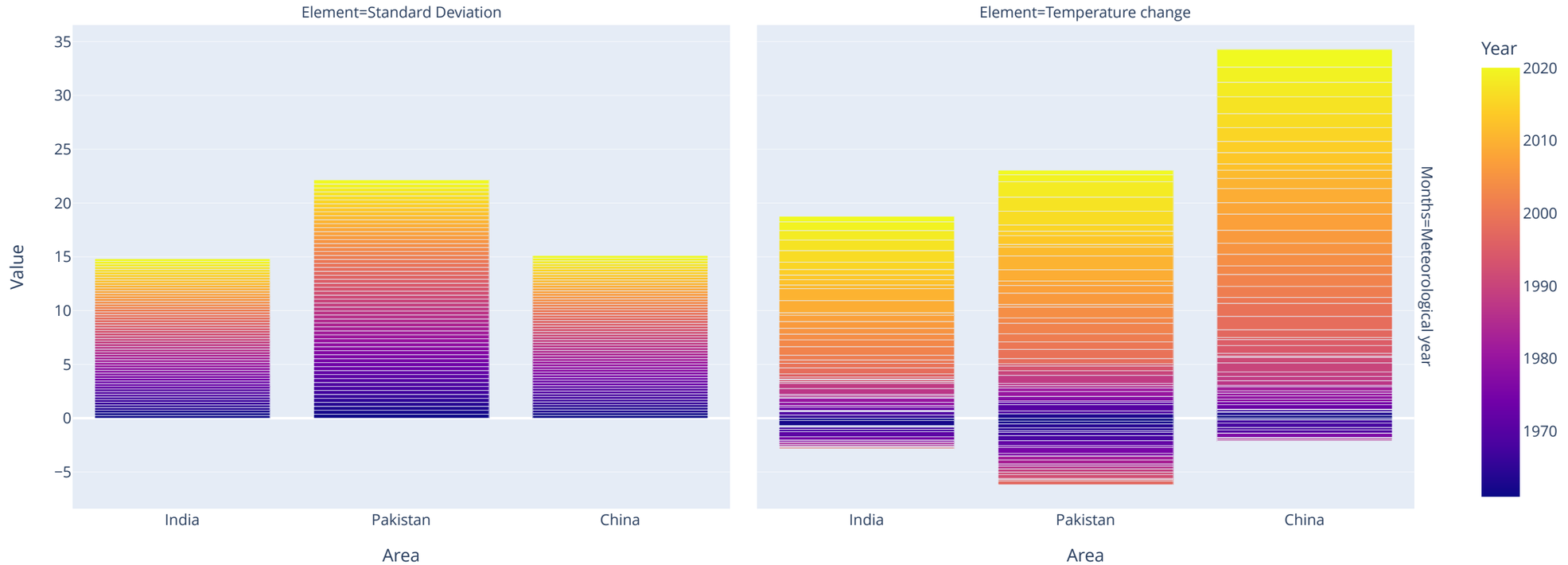
In [ ]:
```
# Draw boxplot
plt.figure(figsize=(26,6))
sns.boxplot(x="Year", y="Value", data=ass2)
```

Out[ ]: <AxesSubplot:xlabel='Year', ylabel='Value'>



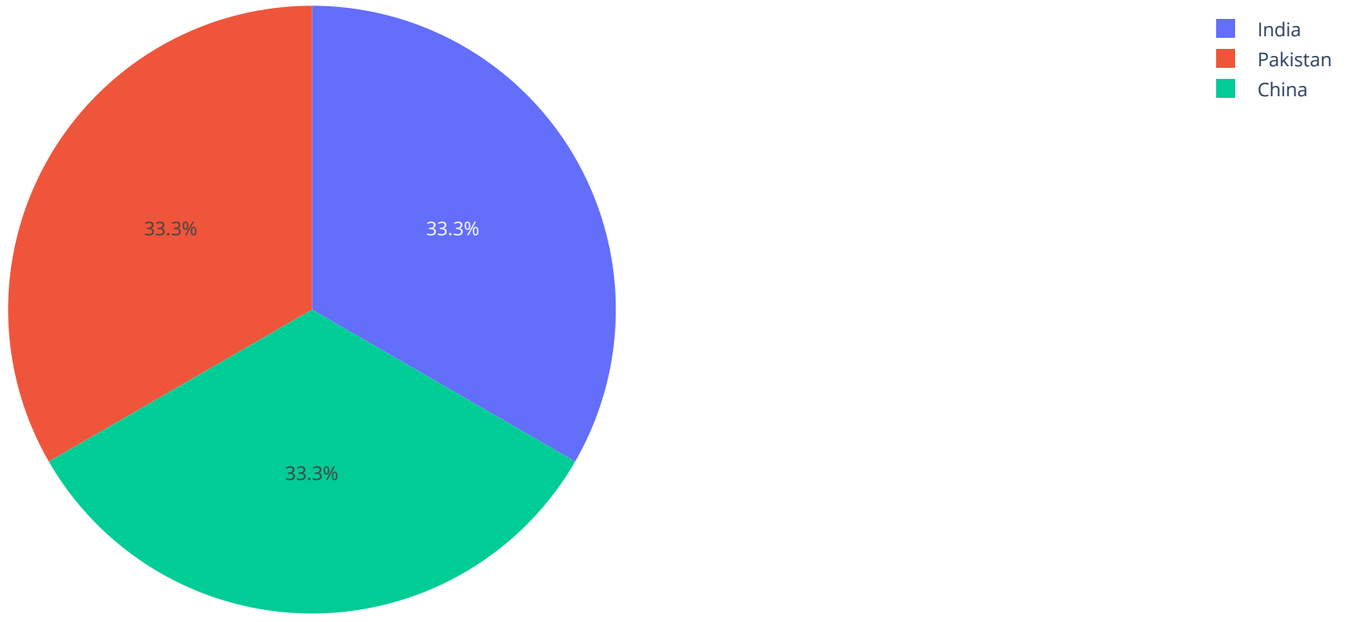
In [ ]:
```
changes = pd.read_csv("temp_change.csv")
```

In [ ]:
```
import plotly.express as px
df = pd.read_csv("temp_change.csv")
fig = px.bar(df, x="Area", y="Value", color="Year", barmode="group", facet_row="Months", facet_col="Element")
fig.show()
```



In [ ]:
```
import plotly.express as px

df = pd.read_csv("temp_change.csv")
fig = px.pie(df, values='Year', names='Area')
fig.update_traces(textposition='inside')
fig.update_layout(uniformtext_minsize=12, uniformtext_mode='hide')
fig.show()
```



In [ ]:
```
# import plotly.express as px
# df = pd.read_csv("temp_change.csv")
# px.scatter(df, x="Element", y="Value", animation_frame="Year", animation_group="Area",
#            size="Flag", color="Months", hover_name="Area",
#            log_x=True, size_max=55, range_x=[100,100000], range_y=[25,90])

# NOT WORKING
```

PANDAS

In [ ]:
```
import pandas as pd
import numpy as np
```

In [ ]:
```
# Object Creation
d = pd.Series([3,5,np.nan,8,9])
d
```

Out[ ]:
```
0    3.0
1    5.0
2    NaN
3    8.0
4    9.0
dtype: float64
```

In [ ]:
```
dates = pd.date_range("20220316",periods=6)
dates
```

Out[ ]:
```
DatetimeIndex(['2022-03-16', '2022-03-17', '2022-03-18', '2022-03-19',
               '2022-03-20', '2022-03-21'],
              dtype='datetime64[ns]', freq='D')
```

In [ ]:
```
dates = pd.date_range("20220301",periods=20)
df = pd.DataFrame(np.random.randn(20,5), index=dates, columns=list("ABCDE"))
df.head()
```

Out[ ]:

	A	B	C	D	E
2022-03-01	-0.104906	-0.273983	1.337041	0.559453	-1.612230
2022-03-02	0.895248	0.036721	0.276030	-1.606460	0.481814
2022-03-03	1.438060	1.002553	0.812543	0.985676	1.362653
2022-03-04	-1.679950	-0.114346	0.916151	1.377334	0.434219
2022-03-05	0.512177	-0.269029	-1.097855	0.420015	0.212026

In [ ]:
```
df2 = pd.DataFrame(
    {
        "A": 1.0,
        "B": pd.Timestamp("20220923"),
        "C": pd.Series(1, index=list(range(4)), dtype="float32"),
        "D": np.array([3] * 4, dtype="int32"),
        "E": pd.Categorical(["girl", "women", "girl", "women"]),
        "F": "female",

    }
)

df2
```

Out[ ]:

	A	B	C	D	E	F
0	1.0	2022-09-23	1.0	3	girl	female
1	1.0	2022-09-23	1.0	3	women	female
2	1.0	2022-09-23	1.0	3	girl	female
3	1.0	2022-09-23	1.0	3	women	female

In [ ]:
```
df.index
```

Out[ ]:
```
DatetimeIndex(['2022-03-01', '2022-03-02', '2022-03-03', '2022-03-04',
               '2022-03-05', '2022-03-06', '2022-03-07', '2022-03-08',
               '2022-03-09', '2022-03-10', '2022-03-11', '2022-03-12',
               '2022-03-13', '2022-03-14', '2022-03-15', '2022-03-16',
               '2022-03-17', '2022-03-18', '2022-03-19', '2022-03-20'],
              dtype='datetime64[ns]', freq='D')
```

In [ ]:
```
df.sort_index(axis=1, ascending=True)
```

Out[ ]:

	A	B	C	D	E
2022-03-01	-0.104906	-0.273983	1.337041	0.559453	-1.612230
2022-03-02	0.895248	0.036721	0.276030	-1.606460	0.481814
2022-03-03	1.438060	1.002553	0.812543	0.985676	1.362653
2022-03-04	-1.679950	-0.114346	0.916151	1.377334	0.434219
2022-03-05	0.512177	-0.269029	-1.097855	0.420015	0.212026
2022-03-06	-0.829965	2.465498	-1.471057	-0.488825	-1.020155
2022-03-07	0.439787	1.194357	-0.148008	1.278895	0.212569
2022-03-08	-1.406589	-0.575421	-0.070543	0.758489	0.855842
2022-03-09	-0.235417	0.699986	0.990638	-0.919004	0.608564
2022-03-10	0.744770	1.169245	-0.386819	-0.207121	0.075562
2022-03-11	0.268807	1.273896	0.280964	0.606315	-1.102610
2022-03-12	-0.481200	0.074302	-1.045255	0.208357	0.587097
2022-03-13	3.133341	0.895814	-0.613201	0.844418	0.492308
2022-03-14	-0.589419	0.128729	1.249411	-0.472051	-0.385398
2022-03-15	-0.667314	2.373205	1.297170	-1.333298	1.538700
2022-03-16	0.076712	-0.554930	-0.224276	0.427035	0.870743
2022-03-17	0.389891	-0.013000	0.066541	-2.080708	1.422983
2022-03-18	-0.643780	0.131111	0.323243	0.557844	0.035427
2022-03-19	0.295693	-1.787544	-0.545631	1.488240	0.135001
2022-03-20	-0.653825	-0.813783	-2.735544	-1.762935	0.812740

In [ ]: