07_pandas_exercise_solution

October 18, 2020

1 Pandas Übung

1.1 Import

- Importiere `pandas` als `pd`
- Importiere `numpy` als `np`

[78]:
```
import pandas as pd
import numpy as np
```

Lade aus der Datei `04_pandas-excelbsp.xlsx` das Sheet `BL_7-Tage-Inzidenz` als `pd.DataFrame` ein. Verwende das Argument `index_col=0`, um die Bundesländer als Index zu setzen für die Zeilen.

[79]:
```
df = pd.read_excel('04_pandas_excelbsp.xlsx', sheet_name='BL_7-Tage-Inzidenz',␣
↪index_col=0)
```

[80]:
```
# Lasse dir die ersten 5 Zeilen anzeigen (Tipp: head)
df.head()
```

[80]:

Bundesland	2020-05-06	2020-05-07	2020-05-08	2020-05-09	2020-05-10	\
Baden-Württemberg	8.202695	7.037334	6.007480	6.377866	6.974097	
Bayern	10.078979	9.528383	9.038963	9.352497	8.778959	
Berlin	7.363637	7.070158	7.256917	7.470356	7.176878	
Brandenburg	4.976279	4.617987	4.657797	5.215140	4.697607	
Bremen	17.423490	18.448402	20.498224	22.694462	24.890701	

Bundesland	2020-05-11	2020-05-12	2020-05-13	2020-05-14	2020-05-15	\
Baden-Württemberg	6.865692	6.992165	6.721151	6.910861	6.549508	
Bayern	8.801901	9.161318	9.130729	8.526602	7.815415	
Berlin	6.616601	5.282609	4.535573	4.162056	4.108696	
Brandenburg	4.697607	4.180074	2.866337	1.990512	2.189563	
Bremen	24.158621	22.108799	23.426542	19.619729	19.034065	

Bundesland	…	2020-10-16	2020-10-17	2020-10-18	2020-10-19	\
Baden-Württemberg	…	38.070721	NaN	NaN	NaN	

```
Bayern                …   35.406424         NaN         NaN         NaN
Berlin                …   73.852205         NaN         NaN         NaN
Brandenburg           …   20.302209         NaN         NaN         NaN
Bremen                …   73.106068         NaN         NaN         NaN

                    2020-10-20  2020-10-21  2020-10-22  2020-10-23  2020-10-24  \
Bundesland
Baden-Württemberg          NaN         NaN         NaN         NaN         NaN
Bayern                     NaN         NaN         NaN         NaN         NaN
Berlin                     NaN         NaN         NaN         NaN         NaN
Brandenburg                NaN         NaN         NaN         NaN         NaN
Bremen                     NaN         NaN         NaN         NaN         NaN

                    2020-10-25
Bundesland
Baden-Württemberg          NaN
Bayern                     NaN
Berlin                     NaN
Brandenburg                NaN
Bremen                     NaN

[5 rows x 173 columns]
```

[81]:
```
# Lasse dir alle Indizenzen für den 13. Oktober (2020-10-13) ausgeben
df['2020-10-13']
```

[81]:
```
Bundesland
Baden-Württemberg         31.566447
Bayern                    27.398644
Berlin                    60.471602
Brandenburg               11.657909
Bremen                    70.757279
Hamburg                   27.770966
Hessen                    34.064452
Mecklenburg-Vorpommern     7.213311
Niedersachsen             22.217752
Nordrhein-Westfalen       39.660736
Rheinland-Pfalz           22.887694
Saarland                  26.548126
Sachsen                   21.046319
Sachsen-Anhalt             7.654519
Schleswig-Holstein        11.777780
Thüringen                 12.046623
Name: 2020-10-13 00:00:00, dtype: float64
```

[82]:
```
# Lasse dir das Bundesland ausgeben, in dem die höchste Inzidenz geherrscht hat␣
 ↪am 13. Oktober
```

```
# Tipp: idxmax()
df['2020-10-13'].idxmax()
```

[82]:
```
'Bremen'
```

[83]:
```
# Lasse dir für Baden-Württemberg die Zahlen von 01. Oktober bis einschließlich 15. Oktober ausgeben
# Tipp: df.loc[...][...:...]
df.loc['Baden-Württemberg']['2020-10-01':'2020-10-15']
```

[83]:
```
2020-10-01    14.237276
2020-10-02    15.050319
2020-10-03    13.379065
2020-10-04    15.646550
2020-10-05    15.917564
2020-10-06    16.423457
2020-10-07    17.670122
2020-10-08    20.638907
2020-10-09    22.503706
2020-10-10    23.611775
2020-10-11    25.845929
2020-10-12    28.674658
2020-10-13    31.566447
2020-10-14    32.386238
2020-10-15    34.575349
Name: Baden-Württemberg, dtype: float64
```

[84]:
```
# Gruppiere nach Wochentagen (Mo-Fr), und bilde den jeweiligen Mittelwert über alle Observationen an dem Wochentag für jedes Bundesland
# Tipp: df.columns.weekday, axis=1, mean
df.groupby(df.columns.weekday, axis=1).mean()
```

[84]:

Bundesland	0	1	2	3	4
Baden-Württemberg	7.326868	7.716542	7.837957	7.928278	8.166675
Bayern	8.962397	9.158606	9.185912	9.622373	9.746626
Berlin	13.449923	13.172378	13.400860	13.602441	14.017188
Brandenburg	2.606341	2.585652	2.855182	3.029291	3.090410
Bremen	13.414321	13.771129	14.173118	14.544654	15.082455
Hamburg	7.743725	7.798201	7.846529	7.679646	8.038808
Hessen	8.967048	9.142380	9.226510	9.408601	9.866930
Mecklenburg-Vorpommern	1.491201	1.842421	1.882231	2.081940	2.076797
Niedersachsen	5.731704	6.095722	5.863141	5.877328	6.311271
Nordrhein-Westfalen	10.539832	10.945355	10.975779	11.152956	11.488488
Rheinland-Pfalz	5.850875	5.930647	6.033254	6.183597	6.311061
Saarland	4.643860	4.854604	5.528921	5.628729	5.641997
Sachsen	3.413033	3.444043	3.467127	3.786833	3.804332

```
Sachsen-Anhalt            2.467123   2.488683   2.209317   2.397810   2.411145
Schleswig-Holstein        3.527600   3.661075   3.777471   3.803906   3.852703
Thüringen                 3.929650   3.925918   4.060234   4.201889   4.235492
```

```
                                5          6
Bundesland
Baden-Württemberg        7.012084   6.971357
Bayern                   8.860385   8.655081
Berlin                  12.147638  12.989505
Brandenburg              2.526994   2.490515
Bremen                  12.501713  13.382308
Hamburg                  7.264307   7.625467
Hessen                   8.451201   8.833253
Mecklenburg-Vorpommern   1.658749   1.515526
Niedersachsen            5.675481   5.528455
Nordrhein-Westfalen     10.133877  10.276137
Rheinland-Pfalz          5.473392   5.325406
Saarland                 4.054315   4.498620
Sachsen                  3.209200   3.256207
Sachsen-Anhalt           2.094369   2.348774
Schleswig-Holstein       3.584672   3.578639
Thüringen                3.696079   3.787640
```

[85]:
```
# Verwende das Ergebnis des letzten Schritts und berechne den Mittelwert über␣
 ↪alle Bundesländer für die jeweiligen Wochentage
# Tipp: groupby, mean
df.groupby(df.columns.weekday, axis=1).mean().mean()
```

[85]:
```
0    6.504094
1    6.658335
2    6.770221
3    6.933142
4    7.133899
5    6.146529
6    6.316431
dtype: float64
```

[86]:
```
# Logarithmiere alle Werte im Datensatz und speichere ihn als df_log
# Tipp: apply, np.log
df_log = df.apply(np.log, axis=1)
```

[87]:
```
# Berechne nun die Differenz der logarithmierten Werte von einem auf den␣
 ↪nächsten Tag
# Tipp: diff, axis=1
df_logdiff = df.diff(axis=1)
```

[88]:
```
# Berechne nun den Mittelwert für jedes Bundesland über alle Beobachtungen
# Tipp: mean, axis=1, skipna=True
df_logdiff.mean(axis=1, skipna=True)
```

[88]:
```
Bundesland
Baden-Württemberg          0.183239
Bayern                     0.155383
Berlin                     0.407905
Brandenburg                0.094024
Bremen                     0.341611
Hamburg                    0.168609
Hessen                     0.243075
Mecklenburg-Vorpommern     0.077070
Niedersachsen              0.140488
Nordrhein-Westfalen        0.255480
Rheinland-Pfalz            0.133153
Saarland                   0.216547
Sachsen                    0.133219
Sachsen-Anhalt             0.035033
Schleswig-Holstein         0.045352
Thüringen                  0.051447
dtype: float64
```

[89]:
```
# Berechne nun den Median für jedes Bundesland über alle Beobachtungen
# Tipp: median, axis=1, skipna=True
# df_logdiff.median(axis=1, skipna=True)
```

Glückwunsch, du hast es geschafft!