Geodatenanalyse 2

Termin: Big Data 1 - Modul 2

Einlesen und Analyse von Zeitreihen

Ca. 20-30 Minuten

Inhalt

- Vektorisierung von Zeitreihen
- Erstellung von Zeitreihen
- Einlesen von Zeitreihen
- Indizierung von Datum und Zeit
- Rechnen mit Datum und Zeit
- Konvertierung in einen Index
- Konvertierung von Datum und Zeit in Fließkommazahlen

In [1]:
```
import datetime as dt
import numpy as np
import pandas as pd
```

Vektorisierung? *Pandas* to the rescue ...

- Die vorigen Beispiele sind nicht vektororientiert
- Für Zeitreihen muss man Schleifen verwenden
- Dieses Problem wurde von *Pandas* gelöst

Erstellung von Zeitreihen

In [2]:
```
datetimes = pd.date_range(start='2021-04-01', end='2021-04-30', freq='D')
datetimes
```

Out[2]:
```
DatetimeIndex(['2021-04-01', '2021-04-02', '2021-04-03', '2021-04-04',
               '2021-04-05', '2021-04-06', '2021-04-07', '2021-04-08',
               '2021-04-09', '2021-04-10', '2021-04-11', '2021-04-12',
               '2021-04-13', '2021-04-14', '2021-04-15', '2021-04-16',
               '2021-04-17', '2021-04-18', '2021-04-19', '2021-04-20',
               '2021-04-21', '2021-04-22', '2021-04-23', '2021-04-24',
               '2021-04-25', '2021-04-26', '2021-04-27', '2021-04-28',
               '2021-04-29', '2021-04-30'],
              dtype='datetime64[ns]', freq='D')
```

Erstellung über Start und Anzahl der Einträge

In [3]:
```
datetimes = pd.date_range(start='2021-04-01', periods=100, freq='3D')
datetimes
```

Out[3]:
```
DatetimeIndex(['2021-04-01', '2021-04-04', '2021-04-07', '2021-04-10',
               '2021-04-13', '2021-04-16', '2021-04-19', '2021-04-22',
               '2021-04-25', '2021-04-28', '2021-05-01', '2021-05-04',
               '2021-05-07', '2021-05-10', '2021-05-13', '2021-05-16',
               '2021-05-19', '2021-05-22', '2021-05-25', '2021-05-28',
               '2021-05-31', '2021-06-03', '2021-06-06', '2021-06-09',
               '2021-06-12', '2021-06-15', '2021-06-18', '2021-06-21',
               '2021-06-24', '2021-06-27', '2021-06-30', '2021-07-03',
               '2021-07-06', '2021-07-09', '2021-07-12', '2021-07-15',
               '2021-07-18', '2021-07-21', '2021-07-24', '2021-07-27',
               '2021-07-30', '2021-08-02', '2021-08-05', '2021-08-08',
               '2021-08-11', '2021-08-14', '2021-08-17', '2021-08-20',
               '2021-08-23', '2021-08-26', '2021-08-29', '2021-09-01',
               '2021-09-04', '2021-09-07', '2021-09-10', '2021-09-13',
               '2021-09-16', '2021-09-19', '2021-09-22', '2021-09-25',
               '2021-09-28', '2021-10-01', '2021-10-04', '2021-10-07',
               '2021-10-10', '2021-10-13', '2021-10-16', '2021-10-19',
               '2021-10-22', '2021-10-25', '2021-10-28', '2021-10-31',
               '2021-11-03', '2021-11-06', '2021-11-09', '2021-11-12',
               '2021-11-15', '2021-11-18', '2021-11-21', '2021-11-24',
               '2021-11-27', '2021-11-30', '2021-12-03', '2021-12-06',
               '2021-12-09', '2021-12-12', '2021-12-15', '2021-12-18',
               '2021-12-21', '2021-12-24', '2021-12-27', '2021-12-30',
               '2022-01-02', '2022-01-05', '2022-01-08', '2022-01-11',
               '2022-01-14', '2022-01-17', '2022-01-20', '2022-01-23'],
              dtype='datetime64[ns]', freq='3D')
```

Wichtig hier ist die Syntax für das Schlüsselwort *freq*:

Alias	Description
B	business day frequency
C	custom business day frequency
D	calendar day frequency
W	weekly frequency
M	month end frequency

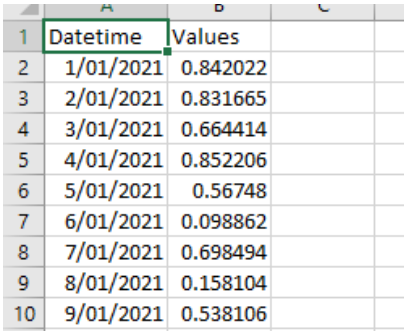
> Siehe auch die Referenz für Pandas date_range()

Das Einlesen von Zeitreihen

- Meistens liegen Datum und Zeit in einem Datencontainer (z.B. Excel oder CSV) vor
- Diese sind entweder als Zeichenkette oder Fließkommazahl abgespeichert
- Hier muss eine richtige Umwandlung erfolgen

Automatisch erkennbare Formate

Beispiel: Datum



	A	B	C
1	Datetime	Values	
2	1/01/2021	0.842022	
3	2/01/2021	0.831665	
4	3/01/2021	0.664414	
5	4/01/2021	0.852206	
6	5/01/2021	0.56748	
7	6/01/2021	0.098862	
8	7/01/2021	0.698494	
9	8/01/2021	0.158104	
10	9/01/2021	0.538106	

In [4]:
```
data = pd.read_csv("data/date_int.csv")
data.loc[:, 'Date'].values
```

Out[4]:
```
array(['1/01/2021', '2/01/2021', '3/01/2021', '4/01/2021', '5/01/2021',
       '6/01/2021', '7/01/2021', '8/01/2021', '9/01/2021', '10/01/2021',
       '11/01/2021', '12/01/2021', '13/01/2021', '14/01/2021',
       '15/01/2021', '16/01/2021', '17/01/2021', '18/01/2021',
       '19/01/2021', '20/01/2021', '21/01/2021', '22/01/2021',
       '23/01/2021', '24/01/2021', '25/01/2021', '26/01/2021',
       '27/01/2021', '28/01/2021', '29/01/2021', '30/01/2021'],
      dtype=object)
```

In [5]:
```
data['New'] = pd.to_datetime(data['Date'], dayfirst=True)
data.loc[:, 'New'].values
```

Out[5]:
```
array(['2021-01-01T00:00:00.000000000', '2021-01-02T00:00:00.000000000',
       '2021-01-03T00:00:00.000000000', '2021-01-04T00:00:00.000000000',
       '2021-01-05T00:00:00.000000000', '2021-01-06T00:00:00.000000000',
       '2021-01-07T00:00:00.000000000', '2021-01-08T00:00:00.000000000',
       '2021-01-09T00:00:00.000000000', '2021-01-10T00:00:00.000000000',
       '2021-01-11T00:00:00.000000000', '2021-01-12T00:00:00.000000000',
       '2021-01-13T00:00:00.000000000', '2021-01-14T00:00:00.000000000',
       '2021-01-15T00:00:00.000000000', '2021-01-16T00:00:00.000000000',
       '2021-01-17T00:00:00.000000000', '2021-01-18T00:00:00.000000000',
       '2021-01-19T00:00:00.000000000', '2021-01-20T00:00:00.000000000',
       '2021-01-21T00:00:00.000000000', '2021-01-22T00:00:00.000000000',
       '2021-01-23T00:00:00.000000000', '2021-01-24T00:00:00.000000000',
       '2021-01-25T00:00:00.000000000', '2021-01-26T00:00:00.000000000',
       '2021-01-27T00:00:00.000000000', '2021-01-28T00:00:00.000000000',
       '2021-01-29T00:00:00.000000000', '2021-01-30T00:00:00.000000000'],
      dtype='datetime64[ns]')
```

In [6]:
```
data = pd.read_csv("data/date_int.csv", parse_dates=[0])
data.loc[:, 'Date'].values
```

```
C:\Users\gcr133\AppData\Local\Temp\1\ipykernel_13624\2548043229.py:1: UserWarning:
Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) was specifie
d. This may lead to inconsistently parsed dates! Specify a format to ensure consis
tent parsing.
  data = pd.read_csv("data/date_int.csv", parse_dates=[0])
```

Out[6]:
```
array(['2021-01-01T00:00:00.000000000', '2021-02-01T00:00:00.000000000',
       '2021-03-01T00:00:00.000000000', '2021-04-01T00:00:00.000000000',
       '2021-05-01T00:00:00.000000000', '2021-06-01T00:00:00.000000000',
       '2021-07-01T00:00:00.000000000', '2021-08-01T00:00:00.000000000',
       '2021-09-01T00:00:00.000000000', '2021-10-01T00:00:00.000000000',
       '2021-11-01T00:00:00.000000000', '2021-12-01T00:00:00.000000000',
       '2021-01-13T00:00:00.000000000', '2021-01-14T00:00:00.000000000',
       '2021-01-15T00:00:00.000000000', '2021-01-16T00:00:00.000000000',
       '2021-01-17T00:00:00.000000000', '2021-01-18T00:00:00.000000000',
       '2021-01-19T00:00:00.000000000', '2021-01-20T00:00:00.000000000',
       '2021-01-21T00:00:00.000000000', '2021-01-22T00:00:00.000000000',
```

```
       '2021-01-23T00:00:00.000000000', '2021-01-24T00:00:00.000000000',
       '2021-01-25T00:00:00.000000000', '2021-01-26T00:00:00.000000000',
       '2021-01-27T00:00:00.000000000', '2021-01-28T00:00:00.000000000',
       '2021-01-29T00:00:00.000000000', '2021-01-30T00:00:00.000000000'],
```

Achtung Formatfalle: Amerikaner haben das Datumsformat MM/DD/YYYY anstelle von DD/MM/YYYY. Das führt manchmal zu schwierig erkennbaren Einlesefehlern! Hier gibt es das Schlüsselwort `dayfirst=True`

Beispiel: Getrennte Datum und Zeit

	A	B	C	D
1	Date	Time	Values	
2	1/01/2021	00:00:00	0.381004	
3	1/01/2021	00:10:00	0.369205	
4	1/01/2021	00:20:00	0.1429	
5	1/01/2021	00:30:00	0.777587	
6	1/01/2021	00:40:00	0.883736	
7	1/01/2021	00:50:00	0.673375	
8	1/01/2021	01:00:00	0.390332	
9	1/01/2021	01:10:00	0.498278	
10	1/01/2021	01:20:00	0.926908	
11	1/01/2021	01:30:00	0.750456	
12	1/01/2021	01:40:00	0.429851	

In [7]:
```
data = pd.read_csv("data/date-time_int.csv", parse_dates=[[0,1]])
data.loc[:, 'Date_Time'].values
```

Out[7]:
```
array(['2021-01-01T00:00:00.000000000', '2021-01-01T00:10:00.000000000',
       '2021-01-01T00:20:00.000000000', '2021-01-01T00:30:00.000000000',
       '2021-01-01T00:40:00.000000000', '2021-01-01T00:50:00.000000000',
       '2021-01-01T01:00:00.000000000', '2021-01-01T01:10:00.000000000',
       '2021-01-01T01:20:00.000000000', '2021-01-01T01:30:00.000000000',
       '2021-01-01T01:40:00.000000000', '2021-01-01T01:50:00.000000000',
       '2021-01-01T02:00:00.000000000', '2021-01-01T02:10:00.000000000',
       '2021-01-01T02:20:00.000000000', '2021-01-01T02:30:00.000000000',
       '2021-01-01T02:40:00.000000000', '2021-01-01T02:50:00.000000000',
       '2021-01-01T03:00:00.000000000', '2021-01-01T03:10:00.000000000',
       '2021-01-01T03:20:00.000000000', '2021-01-01T03:30:00.000000000',
       '2021-01-01T03:40:00.000000000', '2021-01-01T03:50:00.000000000',
       '2021-01-01T04:00:00.000000000', '2021-01-01T04:10:00.000000000',
       '2021-01-01T04:20:00.000000000', '2021-01-01T04:30:00.000000000',
       '2021-01-01T04:40:00.000000000', '2021-01-01T04:50:00.000000000'],
      dtype='datetime64[ns]')
```

Beliebige Formate (z.B. deutsches Format)

Beispiel: Datum und Zeit

	A	B	C
1	Datum & Zeit	Values	
2	01.01.2021 00:00	0.381004	
3	01.01.2021 00:10	0.369205	
4	01.01.2021 00:20	0.1429	
5	01.01.2021 00:30	0.777587	
6	01.01.2021 00:40	0.883736	
7	01.01.2021 00:50	0.673375	
8	01.01.2021 01:00	0.390332	
9	01.01.2021 01:10	0.498278	
10	01.01.2021 01:20	0.926908	

11	01.01.2021 01:30	0.750456
12	01.01.2021 01:40	0.429851
13	01.01.2021 01:50	0.177479
14	01.01.2021 02:00	0.553873

Erste Möglichkeit

Mit einer Hilfsfunktion ...

In [8]:
```
# need to construct a helper function ...
from datetime import datetime
custom_date_parser = lambda x: datetime.strptime(x, "%d.%m.%Y %H:%M")

data = pd.read_csv("data/date-time_de.csv", parse_dates=[0], date_parser=custom_da
data.loc[:, 'Datum & Zeit'].values
```

Out[8]:
```
array(['2021-01-01T00:00:00.000000000', '2021-01-01T00:10:00.000000000',
       '2021-01-01T00:20:00.000000000', '2021-01-01T00:30:00.000000000',
       '2021-01-01T00:40:00.000000000', '2021-01-01T00:50:00.000000000',
       '2021-01-01T01:00:00.000000000', '2021-01-01T01:10:00.000000000',
       '2021-01-01T01:20:00.000000000', '2021-01-01T01:30:00.000000000',
       '2021-01-01T01:40:00.000000000', '2021-01-01T01:50:00.000000000',
       '2021-01-01T02:00:00.000000000', '2021-01-01T02:10:00.000000000',
       '2021-01-01T02:20:00.000000000', '2021-01-01T02:30:00.000000000',
       '2021-01-01T02:40:00.000000000', '2021-01-01T02:50:00.000000000',
       '2021-01-01T03:00:00.000000000', '2021-01-01T03:10:00.000000000',
       '2021-01-01T03:20:00.000000000', '2021-01-01T03:30:00.000000000',
       '2021-01-01T03:40:00.000000000', '2021-01-01T03:50:00.000000000',
       '2021-01-01T04:00:00.000000000', '2021-01-01T04:10:00.000000000',
       '2021-01-01T04:20:00.000000000', '2021-01-01T04:30:00.000000000',
       '2021-01-01T04:40:00.000000000', '2021-01-01T04:50:00.000000000'],
      dtype='datetime64[ns]')
```

Zweite Möglichkeit

In zwei Schritten ...

In [9]:
```
datum = pd.read_csv("data/date-time_de.csv")
datum['DE'] = pd.to_datetime(datum['Datum & Zeit'], format='%d.%m.%Y %H:%M')
datum['DE'].values
```

Out[9]:
```
array(['2021-01-01T00:00:00.000000000', '2021-01-01T00:10:00.000000000',
       '2021-01-01T00:20:00.000000000', '2021-01-01T00:30:00.000000000',
       '2021-01-01T00:40:00.000000000', '2021-01-01T00:50:00.000000000',
       '2021-01-01T01:00:00.000000000', '2021-01-01T01:10:00.000000000',
       '2021-01-01T01:20:00.000000000', '2021-01-01T01:30:00.000000000',
       '2021-01-01T01:40:00.000000000', '2021-01-01T01:50:00.000000000',
       '2021-01-01T02:00:00.000000000', '2021-01-01T02:10:00.000000000',
       '2021-01-01T02:20:00.000000000', '2021-01-01T02:30:00.000000000',
       '2021-01-01T02:40:00.000000000', '2021-01-01T02:50:00.000000000',
       '2021-01-01T03:00:00.000000000', '2021-01-01T03:10:00.000000000',
       '2021-01-01T03:20:00.000000000', '2021-01-01T03:30:00.000000000',
       '2021-01-01T03:40:00.000000000', '2021-01-01T03:50:00.000000000',
       '2021-01-01T04:00:00.000000000', '2021-01-01T04:10:00.000000000',
       '2021-01-01T04:20:00.000000000', '2021-01-01T04:30:00.000000000',
       '2021-01-01T04:40:00.000000000', '2021-01-01T04:50:00.000000000'],
      dtype='datetime64[ns]')
```

Indizierung von Datum und Zeit

Die Indizierung funktioniert genauso wie mit *label* oder *location*

Man beachte hier die einfache Verwendung von *strings* in der Vorgabe!

In [10]:
```
data = pd.read_csv("data/date_int.csv")
data['Datetime'] = pd.to_datetime(data.loc[:, 'Date'], dayfirst=True)

index = data['Datetime'] > '2021-01-13'
data[index]
```

Out[10]:

	Date	Values	Datetime
13	14/01/2021	0.198333	2021-01-14
14	15/01/2021	0.900442	2021-01-15
15	16/01/2021	0.610314	2021-01-16
16	17/01/2021	0.815259	2021-01-17
17	18/01/2021	0.896095	2021-01-18
18	19/01/2021	0.917005	2021-01-19
19	20/01/2021	0.683041	2021-01-20
20	21/01/2021	0.002168	2021-01-21
21	22/01/2021	0.973884	2021-01-22
22	23/01/2021	0.712743	2021-01-23
23	24/01/2021	0.638000	2021-01-24
24	25/01/2021	0.055603	2021-01-25
25	26/01/2021	0.054632	2021-01-26
26	27/01/2021	0.198249	2021-01-27
27	28/01/2021	0.355729	2021-01-28
28	29/01/2021	0.534810	2021-01-29
29	30/01/2021	0.647931	2021-01-30

Natürlich kann man auch andere Datums- und Zeitobjekte verwenden ...

In [11]:
```
data[data['Datetime'] == np.datetime64('2021-01-15')]
```

Out[11]:

	Date	Values	Datetime
14	15/01/2021	0.900442	2021-01-15

Konversion von Datum und Zeit in Zeitreihen

Eingelesene Zeitreihen können in beliebige Datums- und Zeitformate konvertiert werden

In [12]:
```
data['new_format'] = data.loc[:, 'Datetime'].dt.strftime('%d %B %Y')
data['new_format'].values
```

Out[12]:
```
array(['01 January 2021', '02 January 2021', '03 January 2021',
       '04 January 2021', '05 January 2021', '06 January 2021',
       '07 January 2021', '08 January 2021', '09 January 2021',
       '10 January 2021', '11 January 2021', '12 January 2021',
       '13 January 2021', '14 January 2021', '15 January 2021',
       '16 January 2021', '17 January 2021', '18 January 2021',
       '19 January 2021', '20 January 2021', '21 January 2021',
       '22 January 2021', '23 January 2021', '24 January 2021',
       '25 January 2021', '26 January 2021', '27 January 2021',
       '28 January 2021', '29 January 2021', '30 January 2021'],
      dtype=object)
```

Diese können dann auch wieder exportiert werden (siehe auch Geodatenanalyse 1):

In [13]:
```
data.to_csv("data/new_format.csv", index=False)
```

Rechnung mit Datum und Zeit

oftmals muss mit Datum und Zeit gerechnet werden, z.B. um eine Zeitdifferenz ausrechnen zu können

In [14]:
```
datum['Differenz'] = datum['DE'] - datum.loc[10, 'DE']
datum['Differenz'].values
```

Out[14]:
```
array([-6000000000000, -5400000000000, -4800000000000, -4200000000000,
       -3600000000000, -3000000000000, -2400000000000, -1800000000000,
       -1200000000000,  -600000000000,                0,   600000000000,
        1200000000000,  1800000000000,  2400000000000,  3000000000000,
        3600000000000,  4200000000000,  4800000000000,  5400000000000,
        6000000000000,  6600000000000,  7200000000000,  7800000000000,
        8400000000000,  9000000000000,  9600000000000, 10200000000000,
       10800000000000, 11400000000000], dtype='timedelta64[ns]')
```

In [15]:
```
datum['Differenz'].dt.total_seconds().values
```

Out[15]:
```
array([-6000., -5400., -4800., -4200., -3600., -3000., -2400., -1800.,
       -1200.,  -600.,     0.,   600.,  1200.,  1800.,  2400.,  3000.,
        3600.,  4200.,  4800.,  5400.,  6000.,  6600.,  7200.,  7800.,
        8400.,  9000.,  9600., 10200., 10800., 11400.])
```

Konvertierung in einen Index

- Eine Datums- und Zeitreihe kann auch in einen Index verwandelt werden
- Das hilft bei Datenmanipulationen

In [16]:
```
data.set_index('Datetime', inplace=True)
data.index
```

Out[16]:
```
DatetimeIndex(['2021-01-01', '2021-01-02', '2021-01-03', '2021-01-04',
               '2021-01-05', '2021-01-06', '2021-01-07', '2021-01-08',
               '2021-01-09', '2021-01-10', '2021-01-11', '2021-01-12',
               '2021-01-13', '2021-01-14', '2021-01-15', '2021-01-16',
               '2021-01-17', '2021-01-18', '2021-01-19', '2021-01-20',
               '2021-01-21', '2021-01-22', '2021-01-23', '2021-01-24',
               '2021-01-25', '2021-01-26', '2021-01-27', '2021-01-28',
```

```
'2021-01-29', '2021-01-30'],
```

Konvertierung von Datum und Zeit in Fließkommazahlen

Für einige Berechnungen ist es sinnvoll oder nützlich eine Datums- und Zeitreihe in eine Fließkommazahl zu überführen.

Im Folgenden ist ein einfaches Rezept für die Umwandlung in die Excel-basierte Fließkommazahl:

In [17]:
```
td = (datum['DE'] - dt.datetime(1900,1,1)).dt
datum['Numerisch'] = td.days + td.seconds/86400
datum['Numerisch'].values
```

Out[17]:
```
array([44195.        , 44195.00694444, 44195.01388889, 44195.02083333,
       44195.02777778, 44195.03472222, 44195.04166667, 44195.04861111,
       44195.05555556, 44195.0625    , 44195.06944444, 44195.07638889,
       44195.08333333, 44195.09027778, 44195.09722222, 44195.10416667,
       44195.11111111, 44195.11805556, 44195.125     , 44195.13194444,
       44195.13888889, 44195.14583333, 44195.15277778, 44195.15972222,
       44195.16666667, 44195.17361111, 44195.18055556, 44195.1875    ,
       44195.19444444, 44195.20138889])
```

ENDE