10 - Pandas-datetime

March 22, 2023

Table of Contents

1 Working with dates and time in Pandas

[1]:
```
import pandas as pd
import numpy as np
```

1.0.1 1. Convert strings to datetime

[2]:
```
# 1/20/2021 is a string
df = pd.DataFrame({'date': ['1/20/2021', '1/21/2021', '1/22/2021'],
                   'vaccines': [2, 3, 4]})
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   date      3 non-null      object
 1   vaccines  3 non-null      int64
dtypes: int64(1), object(1)
memory usage: 176.0+ bytes
```

[3]:
```
df['date'] = pd.to_datetime(df['date'])
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   date      3 non-null      datetime64[ns]
 1   vaccines  3 non-null      int64
dtypes: datetime64[ns](1), int64(1)
memory usage: 176.0 bytes
```

[4]:
```
df
```

[4]:

	date	vaccines
0	2021-01-20	2
1	2021-01-21	3
2	2021-01-22	4

Day first format

[5]:
```
df = pd.DataFrame({'date': ['3/10/2000', '3/11/2000', '3/12/2000'],
                   'value': [2, 3, 4]})
df['date'] = pd.to_datetime(df['date'], dayfirst=True)
```

[6]:
```
df
```

[6]:

	date	value
0	2000-10-03	2
1	2000-11-03	3
2	2000-12-03	4

Custom format

[7]:
```
df = pd.DataFrame({'date': ['2016-6-10 20:30:0',
                            '2016-7-1 19:45:30',
                            '2013-10-12 4:5:1'],
```

```
                    'value': [2, 3, 4]})
df['date'] = pd.to_datetime(df['date'], format="%Y-%d-%m %H:%M:%S")
```

[8]:
```
df
```

[8]:
```
                 date  value
0 2016-10-06 20:30:00      2
1 2016-01-07 19:45:30      3
2 2013-12-10 04:05:01      4
```

Speedup parsing with infer_datetime_format

[9]:
```
df = pd.DataFrame({'date': ['3/11/2000', '3/12/2000', '3/13/2000'] * 1000 })
df.head()
```

[9]:
```
        date
0  3/11/2000
1  3/12/2000
2  3/13/2000
3  3/11/2000
4  3/12/2000
```

[10]:
```
%timeit pd.to_datetime(df['date'], infer_datetime_format=True)
```

```
985 µs ± 30.1 µs per loop (mean ± std. dev. of 7 runs, 1,000 loops each)
```

[11]:
```
%timeit pd.to_datetime(df['date'], infer_datetime_format=False)
```

```
925 µs ± 6.84 µs per loop (mean ± std. dev. of 7 runs, 1,000 loops each)
```

Handle parsing error

[12]:
```
df = pd.DataFrame({'date': ['3/10/2000', '/11/2000', '3/12/2000'],
                   'value': [2, 3, 4]})
df['date'] = pd.to_datetime(df['date'])
```

[13]:
```
df['date'] = pd.to_datetime(df['date'], errors='ignore')
df
```

[13]:
```
        date  value
0 2000-03-10      2
1 2000-11-01      3
2 2000-03-12      4
```

[14]:
```
df['date'] = pd.to_datetime(df['date'], errors='coerce')
df
```

[14]:
```
        date  value
0 2000-03-10      2
1 2000-11-01      3
2 2000-03-12      4
```

1.0.2 2. Assemble a datetime from multiple columns

[15]:
```
df = pd.DataFrame({'year': [2015, 2016],
                   'month': [2, 3],
                   'day': [4, 5]})

df['date'] = pd.to_datetime(df)
```

[16]:
```
df
```

[16]:
```
   year  month  day       date
0  2015      2    4 2015-02-04
1  2016      3    5 2016-03-05
```

1.0.3 3. Get year, month and day

[17]:
```
df = pd.DataFrame({'name': ['Tom', 'Andy', 'Lucas'],
                   'DoB': ['08-05-1997', '04-28-1996', '12-16-1995']})
df['DoB'] = pd.to_datetime(df['DoB'])
```

[18]:
```
df['year']= df['DoB'].dt.year
df['month']= df['DoB'].dt.month
df['day']= df['DoB'].dt.day
df
```

[18]:
```
    name        DoB  year  month  day
0    Tom 1997-08-05  1997      8    5
1   Andy 1996-04-28  1996      4   28
2  Lucas 1995-12-16  1995     12   16
```

1.0.4 4. Get the week of year, the day of week and leap year

[19]:
```
df['week_of_year'] = df['DoB'].dt.week
df['day_of_week'] = df['DoB'].dt.dayofweek
df['is_leap_year'] = df['DoB'].dt.is_leap_year
df
```

```
/var/folders/bg/jzzhjp857hv08kcqg3jdptcr0000gn/T/ipykernel_38065/3969492849.py:1
: FutureWarning: Series.dt.weekofyear and Series.dt.week have been deprecated.
Please use Series.dt.isocalendar().week instead.
  df['week_of_year'] = df['DoB'].dt.week
```

[19]:
	name	DoB	year	month	day	week_of_year	day_of_week	is_leap_year
0	Tom	1997-08-05	1997	8	5	32	1	False
1	Andy	1996-04-28	1996	4	28	17	6	True
2	Lucas	1995-12-16	1995	12	16	50	5	False

[20]:
```
dw_mapping={
    0: 'Monday',
    1: 'Tuesday',
    2: 'Wednesday',
    3: 'Thursday',
    4: 'Friday',
    5: 'Saturday',
    6: 'Sunday'
}
df['day_of_week_name']=df['DoB'].dt.weekday.map(dw_mapping)
df
```

[20]:
	name	DoB	year	month	day	week_of_year	day_of_week	is_leap_year	day_of_week_name
0	Tom	1997-08-05	1997	8	5	32	1	False	Tuesday
1	Andy	1996-04-28	1996	4	28	17	6	True	Sunday
2	Lucas	1995-12-16	1995	12	16	50	5	False	Saturday

1.0.5 5. Get the age from the date of birth

[21]:
```
today = pd.to_datetime('today')
df['age'] = today.year - df['DoB'].dt.year

df
```

[21]:
	name	DoB	year	month	day	week_of_year	day_of_week	is_leap_year	day_of_week_name	age
0	Tom	1997-08-05	1997	8	5	32	1	False	Tuesday	26
1	Andy	1996-04-28	1996	4	28	17	6	True	Sunday	27
2	Lucas	1995-12-16	1995	12	16	50	5	False	Saturday	28

[22]:
```
# Year difference
today = pd.to_datetime('today')
diff_y = today.year - df['DoB'].dt.year
```

```
# Haven't had birthday
b_md = df['DoB'].apply(lambda x: (x.month,x.day) )
no_birthday = b_md > (today.month,today.day)

df['age'] = diff_y - no_birthday
df
```

[22]:

	name	DoB	year	month	day	week_of_year	day_of_week	is_leap_year	day_of_week_name	age
0	Tom	1997-08-05	1997	8	5	32	1	False	Tuesday	25
1	Andy	1996-04-28	1996	4	28	17	6	True	Sunday	26
2	Lucas	1995-12-16	1995	12	16	50	5	False	Saturday	27

1.0.6 6. Improve performance by setting date column as the index

[ ]:
```
df = pd.read_csv('data/city_sales.csv',parse_dates=['date'])
df.info()
```

[ ]:
```
df = df.set_index(['date'])
```

[ ]:
```
df
```

1.0.7 7. Select data with a specific year and perform aggregation

[ ]:
```
df.loc['2018']
```

[ ]:
```
df.loc['2018','num'].sum()
```

[ ]:
```
df['2018'].groupby('city').sum()
```

1.0.8 8. Select data with a specific month or a specific day of the month

[ ]:
```
df.loc['2018-5']
```

[ ]:
```
df.loc['2018-5-1']
```

[ ]:
```
cond = df.index.month==2
df[cond]
```

1.0.9 9. Select data between two dates

[ ]:
```
df.loc['2016' : '2018']
```

[ ]:
```
df.loc['2018-5-2 10' : '2018-5-2 11' ]
```

[ ]:
```
df.loc['2018-5-2 10:30' : '2018-5-2 10:45' ]
```

[ ]:
```
df.between_time('10:30','10:45')
```

1.0.10 10. Handle missing values

[ ]:
```
df['rolling_sum'] = df.rolling(3).sum()
df.head()
```

[ ]:
```
df['rolling_sum_backfilled'] = df['rolling_sum'].fillna(method='backfill')
df.head()
```

1.0.11 That's it

This is a notebook for the medium article Working with datetime in Pandas DataFrame

Please check out article for instructions

[ ]:
```

```