Dealing with Time

Dates and Time in Python

@cambridgespark

CAMBRIDGE SPARK

Mastering Time

CAMBRIDGE SPARK

Date unit

CAMBRIDGE SPARK

Time unit

CAMBRIDGE SPARK

Datetime

- Combination of Date and Time and also an optional Timezone
- **ISO 8601** represents the standard format for Date and Time
- Example: 2018-10-13T15:53:20

Working with the `datetime` Python module

```
from datetime import datetime, date
new_date = datetime(year=2018, month=10, day=13)
new_date.year
new_date.month
new_date.hour
from dateutil import parser
new_date = parser.parse("13th October 2018")
parser.parse("2018-10-13T15:53:20")
```

CAMBRIDGE SPARK

Hands-on session

python_datetime.ipynb

Numpy Date and Time Support

- More memory efficient representation
- Especially relevant for list of dates

>> See `scripts/datetime_sizer.py`

Working with Numpy `datetime64`

```
import numpy as np
import datetime
np.datetime64("2018-11-03")
np.datetime64("2018-10-03 12:00")
np.array(['2018-11-02', '2018-10-02', '2015-11-03'], dtype='datetime64')
current = np.datetime64(datetime.datetime.now())
import pandas as pd
pd.to_datetime(current).year
np.datetime64("2018-11-03") - np.datetime64("2018-11-01")
np.datetime64('2018-11-03') + np.timedelta64(14, 'D')
np.datetime64('2018-10-03 12:00') + np.timedelta64(6, 'h')
np.datetime64('2018-11-03') + np.arange(10)
```

CAMBRIDGE SPARK

Hands-on session

numpy_datetime.ipynb

CAMBRIDGE SPARK

Pandas Date and Time Support

- Combines ease-of-use of `datetime` and `dateutil` (e.g. accessors)
- Efficient memory representation and manipulation using `numpy`
- Provides integration with pandas Dataframe

CAMBRIDGE SPARK

Pandas Date and Time main classes

Class	**Notes**
`Timestamp`	Represents a datetime (i.e. a point in time)
`DatetimeIndex`	Index of `Timestamp`
`Period`	Represents a time span (i.e. a period of time, fixed-frequency interval)
`PeriodIndex`	Index of `Period`

Working with Pandas Date and Time

```
import pandas as pd
pd.to_datetime("14th of October, 2018")
pd.Timestamp(year=2018, month=10, day=14, hour=12, minute=0, second=30)
datetimes = pd.DatetimeIndex(['2014-07-04', '2014-08-04', '2015-07-04',
'2015-08-04'])
series = pd.Series([10, 4, 14, 30], index=datetimes)
series['2015']
pd.date_range('2015-07-03', '2015-07-10')
pd.date_range('2018 Oct 1', periods = 10, freq = 'W')
```

CAMBRIDGE SPARK

Hands-on session

pandas_datetime.ipynb

What about Timezones?

Timezone are supported in the ISO 8601 standard: **2018-10-14T15:35:35+01:00**

```
london = pd.Timestamp.now(tz="Europe/London")

brussels = london.tz_convert("Europe/Brussels")
```

```
2018-10-14 15:40:01.942971+01:00

2018-10-14 16:40:01.942971+02:00
```

CAMBRIDGE SPARK

Takeaways

- **Python datetime**
    - Native support
    - Inefficient for data analysis for time series
- **Numpy datetime64**
    - Efficient representation
    - Limited set of operations
- **Pandas**
    - Efficient memory representation
    - Integrates with Pandas suite of functions
    - Simple accessors
    - A time series can be represented as a Pandas Series where the index are points in time

CAMBRIDGE SPARK