DSC 201: Data Analysis & Visualization

Time Series

Dr. David Koop
Dates and Times

• What is time to a computer?
  - Can be stored as seconds since Unix Epoch (January 1st, 1970)
• Often useful to break down into minutes, hours, days, months, years…
• Lots of different ways to write time:
  - How could you write "November 29, 2016"?
  - European vs. American ordering…
• What about time zones?
Python Support for Time

• The `datetime` package
  - Has date, time, and datetime classes
  - `.now()` method: the current datetime
  - Can access properties of the time (year, month, seconds, etc.)

• Converting from strings to datetimes:
  - `datetime.strptime`: good for known formats
  - `dateutil.parser.parse`: good for unknown formats

• Converting to strings
  - `str(dt)` or `dt.strftime(<format>)`

• Differences between times
  - `datetime.timedelta`: can get number of days/hours/etc.
  - deal with issues with different length months, etc.
Pandas Support for Datetime

- `pd.to_datetime`:
  - convenience method
  - can convert an entire column to datetime
- Has a NaT to indicate a missing time value (like NaN but for timestamps)
- Stores in a `numpy.datetime64` format
- `pd.Timestamp`: a wrapper for the `datetime64` objects
- Can use time as an index
- Accessing a particular time or checking equivalence allows any string that can be interpreted as a date:
  - `ts['1/10/2011']` or `ts['20110110']`
Generating Date Ranges

- `index = pd.date_range('4/1/2012', '6/1/2012')`
- Can generate based on a number of periods as well
  - `index = pd.date_range('4/1/2012', periods=20)`
- Frequency (`freq`) controls how the range is divided
  - Codes for specifying this (e.g. 4h, D, M)
    - In [90]: `pd.date_range('1/1/2000', '1/3/2000 23:59', freq='4h')`
    - Out[90]:
      - `<class 'pandas.tseries.index.DatetimeIndex'>`
      - `[2000-01-01 00:00:00, ..., 2000-01-03 20:00:00]`
      - Length: 18, Freq: 4H, Timezone: None
  - Can also mix them: '2h30m'
Assignment 5

- http://www.cis.umassd.edu/~dkoop/dsc201/assignment5.html
- Aggregation, resampling, and visualization of time series data
- Last assignment, but likely challenging…start now!
Timedelta

- Compute differences between dates
- Lives in `datetime` module
  ```python
diff = parse_date("1 Jan 2017") - datetime.now().date()
diff.days
  ```
- Also a `pd.Timedelta` object that take strings:
  ```python
datetime.now().date() + pd.Timedelta("4 days")
  ```
- Also, Roll dates using anchored offsets
  ```python
from pandas.tseries.offsets import Day, MonthEnd
now = datetime(2011, 11, 17)
In [107]: now + MonthEnd(2)
Out[107]: Timestamp('2011-12-31 00:00:00')
  ```
Time Zones

- Why?
- Coordinated Universal Time (UTC) is the standard time (basically equivalent to Greenwich Mean Time (GMT))
- Other time zones are UTC +/- a number in [1,12]
- Dartmouth is UTC-5 (aka US/Eastern)
Python, Pandas, and Time Zones

• Time series in pandas are **time zone native**
• The pytz module keeps track of all of the time zone parameters
  - even Daylight Savings Time
• Localize a timestamp using `tz_localize`
  - `ts = pd.Timestamp("1 Dec 2016 12:30 PM")`
  - `ts = ts.tz_localize("US/Eastern")`
• Convert a timestamp using `tz_convert`
  - `ts.tz_convert("Europe/Budapest")`
• Operations involving timestamps from different time zones become UTC
Shifting Data

• Leading or Lagging Data

In [95]: ts = Series(np.random.randn(4),
    index=pd.date_range('1/1/2000', periods=4, freq='M'))

In [96]: ts
Out[96]:
2000-01-31   -0.066748
2000-02-29    0.838639
2000-03-31   -0.117388
2000-04-30   -0.517795
Freq: M, dtype: float64

In [97]: ts.shift(2)
Out[97]:
2000-03-31   -0.066748
2000-04-30    0.838639
2000-05-31   -0.117388
2000-06-30   -0.517795
Freq: M, dtype: float64

In [98]: ts.shift(-2)
Out[98]:
2000-01-31         NaN
2000-02-29   -0.117388
2000-03-31   -0.066748
2000-04-30    0.838639
Freq: M, dtype: float64

• Shifting by time:

In [99]: ts.shift(2, freq='M')
Out[99]:
2000-03-31   -0.066748
2000-04-30    0.838639
2000-05-31   -0.117388
2000-06-30   -0.517795
Freq: M, dtype: float64

[W. McKinney, Python for Data Analysis]
Resampling

• resample method
• Can downsample or upsample data
• If downsample, need to provide method for combining data (mean/sum)
• If upsample, need to fill in missing data
• Interpolation is useful here
Window Functions

• Idea: want to aggregate over a window of time, calculate the answer, and then slide that window ahead. Repeat.

• `rolling`: smooth out data

• In old versions of pandas (like the book uses), this used to be `rolling_count`, `rolling_sum`, `rolling_mean`

• Specify the window size, then an aggregation method

• Can also specify the window

• Result is set to the right edge of window (change with `center=True`)
Food Inspections Example

• Questions:
  - When was a location was last inspected?
  - How many inspections are done per month/year/etc.?
  - Is there any trend in the number of inspections over time?

• Notebook posted on course web page