Lecture 10

Time Series

Alpar Sultan, PhD, Associate professor

- *Timestamps*, specific instants in time
- Fixed *periods*, such as the month January 2007 or the full year 2010
- *Intervals* of time, indicated by a start and end timestamp. Periods can be thought of as special cases of intervals
- Experiment or elapsed time; each timestamp is a measure of time relative to a particular start time (e.g., the diameter of a cookie baking each second since being placed in the oven)

```
In [10]: from datetime import datetime

In [11]: now = datetime.now()

In [12]: now
Out[12]: datetime.datetime(2017, 9, 25, 14, 5, 52, 72973)

In [13]: now.year, now.month, now.day
Out[13]: (2017, 9, 25)

In [14]: delta = datetime(2011, 1, 7) - datetime(2008, 6, 24, 8, 15)

In [15]: delta
Out[15]: datetime.timedelta(926, 56700)

In [16]: delta.days
Out[16]: 926

In [17]: delta.seconds
Out[17]: 56700

In [18]: from datetime import timedelta

In [19]: start = datetime(2011, 1, 7)

In [20]: start + timedelta(12)
Out[20]: datetime.datetime(2011, 1, 19, 0, 0)

In [21]: start - 2 * timedelta(12)
Out[21]: datetime.datetime(2010, 12, 14, 0, 0)
```

Table 10-1. Types in datetime module

Type	Description
`date`	Store calendar date (year, month, day) using the Gregorian calendar
`time`	Store time of day as hours, minutes, seconds, and microseconds
`datetime`	Stores both date and time
`timedelta`	Represents the difference between two `datetime` values (as days, seconds, and microseconds)
`tzinfo`	Base type for storing time zone information

Converting Between String and Datetime

```
In [22]: stamp = datetime(2011, 1, 3)

In [23]: str(stamp)
Out[23]: '2011-01-03 00:00:00'

In [24]: stamp.strftime('%Y-%m-%d')
Out[24]: '2011-01-03'
```

Type	Description
%Y	Four-digit year
%y	Two-digit year
%m	Two-digit month [01, 12]
%d	Two-digit day [01, 31]
%H	Hour (24-hour clock) [00, 23]
%I	Hour (12-hour clock) [01, 12]
%M	Two-digit minute [00, 59]
%S	Second [00, 61] (seconds 60, 61 account for leap seconds)
%w	Weekday as integer [0 (Sunday), 6]

Type	Description
%U	Week number of the year [00, 53]; Sunday is considered the first day of the week, and days before the first Sunday of the year are "week 0"
%W	Week number of the year [00, 53]; Monday is considered the first day of the week, and days before the first Monday of the year are "week 0"
%z	UTC time zone offset as +HHMM or -HHMM; empty if time zone naive
%F	Shortcut for %Y-%m-%d (e.g., 2012-4-18)
%D	Shortcut for %m/%d/%y (e.g., 04/18/12)

```
In [25]: value = '2011-01-03'

In [26]: datetime.strptime(value, '%Y-%m-%d')
Out[26]: datetime.datetime(2011, 1, 3, 0, 0)

In [27]: datestrs = ['7/6/2011', '8/6/2011']

In [28]: [datetime.strptime(x, '%m/%d/%Y') for x in datestrs]
Out[28]:
[datetime.datetime(2011, 7, 6, 0, 0),
 datetime.datetime(2011, 8, 6, 0, 0)]

In [29]: from dateutil.parser import parse

In [30]: parse('2011-01-03')
Out[30]: datetime.datetime(2011, 1, 3, 0, 0)

In [31]: parse('Jan 31, 1997 10:45 PM')
Out[31]: datetime.datetime(1997, 1, 31, 22, 45)

In [32]: parse('6/12/2011', dayfirst=True)
Out[32]: datetime.datetime(2011, 12, 6, 0, 0)

In [33]: datestrs = ['2011-07-06 12:00:00', '2011-08-06 00:00:00']

In [34]: pd.to_datetime(datestrs)
Out[34]: DatetimeIndex(['2011-07-06 12:00:00', '2011-08-06 00:00:00'], dtype='dat
etime64[ns]', freq=None)
```

```
In [35]: idx = pd.to_datetime(datestrs + [None])

In [36]: idx
Out[36]: DatetimeIndex(['2011-07-06 12:00:00', '2011-08-06 00:00:00', 'NaT'], dty
pe='datetime64[ns]', freq=None)

In [37]: idx[2]
Out[37]: NaT

In [38]: pd.isnull(idx)
Out[38]: array([False, False,  True], dtype=bool)
```

Table 10-3. Locale-specific date formatting

Type	Description
%a	Abbreviated weekday name
%A	Full weekday name
%b	Abbreviated month name
%B	Full month name
%c	Full date and time (e.g., 'Tue 01 May 2012 04:20:57 PM')
%p	Locale equivalent of AM or PM
%x	Locale-appropriate formatted date (e.g., in the United States, May 1, 2012 yields '05/01/2012')
%X	Locale-appropriate time (e.g., '04:24:12 PM')

Time Series Basics

```
In [39]: from datetime import datetime

In [40]: dates = [datetime(2011, 1, 2), datetime(2011, 1, 5),
    ....:         datetime(2011, 1, 7), datetime(2011, 1, 8),
    ....:         datetime(2011, 1, 10), datetime(2011, 1, 12)]

In [41]: ts = pd.Series(np.random.randn(6), index=dates)

In [42]: ts
Out[42]:
2011-01-02   -0.204708
2011-01-05    0.478943
2011-01-07   -0.519439
2011-01-08   -0.555730
2011-01-10    1.965781
2011-01-12    1.393406
dtype: float64

In [43]: ts.index
Out[43]:
DatetimeIndex(['2011-01-02', '2011-01-05', '2011-01-07', '2011-01-08',
               '2011-01-10', '2011-01-12'],
              dtype='datetime64[ns]', freq=None)
```

```
In [44]: ts + ts[::2]
Out[44]:
2011-01-02   -0.409415
2011-01-05         NaN
2011-01-07   -1.038877
2011-01-08         NaN
2011-01-10    3.931561
2011-01-12         NaN
dtype: float64

In [45]: ts.index.dtype
Out[45]: dtype('<M8[ns]')

In [46]: stamp = ts.index[0]

In [47]: stamp
Out[47]: Timestamp('2011-01-02 00:00:00')
```

Indexing, Selection, Subsetting

```
In [48]: stamp = ts.index[2]

In [49]: ts[stamp]
Out[49]: -0.51943871505673811
```

As a convenience, you can also pass a string that is interpretable as a date:

```
In [50]: ts['1/10/2011']
Out[50]: 1.9657805725027142

In [51]: ts['20110110']
Out[51]: 1.9657805725027142
```

```
In [52]: longer_ts = pd.Series(np.random.randn(1000),
   ....:                       index=pd.date_range('1/1/2000', periods=1000))

In [53]: longer_ts
Out[53]:
2000-01-01    0.092908
2000-01-02    0.281746
2000-01-03    0.769023
2000-01-04    1.246435
2000-01-05    1.007189
2000-01-06   -1.296221
2000-01-07    0.274992
2000-01-08    0.228913
2000-01-09    1.352917
2000-01-10    0.886429
                ...
2002-09-17   -0.139298
2002-09-18   -1.159926
2002-09-19    0.618965
2002-09-20    1.373890
2002-09-21   -0.983505
Freq: D, Length: 1000, dtype: float64
```

```
In [54]: longer_ts['2001']
Out[54]:
2001-01-01    1.599534
2001-01-02    0.474071
2001-01-03    0.151326
2001-01-04   -0.542173
2001-01-05   -0.475496
2001-01-06    0.106403
2001-01-07   -1.308228
2001-01-08    2.173185
2001-01-09    0.564561
2001-01-10   -0.190481
                ...
2001-12-22    0.000369
2001-12-23    0.900885
2001-12-24   -0.454869
2001-12-25   -0.864547
2001-12-26    1.129120
2001-12-27    0.057874
2001-12-28   -0.433739
2001-12-29    0.092698
2001-12-30   -1.397820
2001-12-31    1.457823
Freq: D, Length: 365, dtype: float64
```

```
In [56]: ts[datetime(2011, 1, 7):]
Out[56]:
2011-01-07   -0.519439
2011-01-08   -0.555730
2011-01-10    1.965781
2011-01-12    1.393406
dtype: float64
```

```
In [57]: ts
Out[57]:
2011-01-02   -0.204708
2011-01-05    0.478943
2011-01-07   -0.519439
2011-01-08   -0.555730
2011-01-10    1.965781
2011-01-12    1.393406
dtype: float64
```

```
In [58]: ts['1/6/2011':'1/11/2011']
Out[58]:
2011-01-07   -0.519439
2011-01-08   -0.555730
2011-01-10    1.965781
dtype: float64
```

```
In [59]: ts.truncate(after='1/9/2011')
Out[59]:
2011-01-02   -0.204708
2011-01-05    0.478943
2011-01-07   -0.519439
2011-01-08   -0.555730
dtype: float64
```

```
In [60]: dates = pd.date_range('1/1/2000', periods=100, freq='W-WED')

In [61]: long_df = pd.DataFrame(np.random.randn(100, 4),
   ....:                        index=dates,
   ....:                        columns=['Colorado', 'Texas',
   ....:                                 'New York', 'Ohio'])

In [62]: long_df.loc['5-2001']
Out[62]:
            Colorado     Texas  New York      Ohio
2001-05-02 -0.006045  0.490094 -0.277186 -0.707213
2001-05-09 -0.560107  2.735527  0.927335  1.513906
2001-05-16  0.538600  1.273768  0.667876 -0.969206
2001-05-23  1.676091 -0.817649  0.050188  1.951312
2001-05-30  3.260383  0.963301  1.201206 -1.852001
```

Time Series with Duplicate Indices

```
In [63]: dates = pd.DatetimeIndex(['1/1/2000', '1/2/2000', '1/2/2000',
   ....:                           '1/2/2000', '1/3/2000'])

In [64]: dup_ts = pd.Series(np.arange(5), index=dates)

In [65]: dup_ts
Out[65]:
2000-01-01    0
2000-01-02    1
2000-01-02    2
2000-01-02    3
2000-01-03    4
dtype: int64

In [66]: dup_ts.index.is_unique
Out[66]: False

In [67]: dup_ts['1/3/2000']  # not duplicated
Out[67]: 4

In [68]: dup_ts['1/2/2000']  # duplicated
Out[68]:
2000-01-02    1
2000-01-02    2
2000-01-02    3
dtype: int64
```

```
In [69]: grouped = dup_ts.groupby(level=0)

In [70]: grouped.mean()
Out[70]:
2000-01-01    0
2000-01-02    2
2000-01-03    4
dtype: int64

In [71]: grouped.count()
Out[71]:
2000-01-01    1
2000-01-02    3
2000-01-03    1
dtype: int64
```

Date Ranges, Frequencies, and Shifting

```
In [72]: ts
Out[72]:
2011-01-02   -0.204708
2011-01-05    0.478943
2011-01-07   -0.519439
2011-01-08   -0.555730
2011-01-10    1.965781
2011-01-12    1.393406
dtype: float64

In [73]: resampler = ts.resample('D')
```

Generating Date Ranges

```
In [74]: index = pd.date_range('2012-04-01', '2012-06-01')

In [75]: index
Out[75]:
DatetimeIndex(['2012-04-01', '2012-04-02', '2012-04-03', '2012-04-04',
               '2012-04-05', '2012-04-06', '2012-04-07', '2012-04-08',
               '2012-04-09', '2012-04-10', '2012-04-11', '2012-04-12',
               '2012-04-13', '2012-04-14', '2012-04-15', '2012-04-16',
               '2012-04-17', '2012-04-18', '2012-04-19', '2012-04-20',
               '2012-04-21', '2012-04-22', '2012-04-23', '2012-04-24',
               '2012-04-25', '2012-04-26', '2012-04-27', '2012-04-28',
               '2012-04-29', '2012-04-30', '2012-05-01', '2012-05-02',
               '2012-05-03', '2012-05-04', '2012-05-05', '2012-05-06',
               '2012-05-07', '2012-05-08', '2012-05-09', '2012-05-10',
               '2012-05-11', '2012-05-12', '2012-05-13', '2012-05-14',
               '2012-05-15', '2012-05-16', '2012-05-17', '2012-05-18',
               '2012-05-19', '2012-05-20', '2012-05-21', '2012-05-22',
               '2012-05-23', '2012-05-24', '2012-05-25', '2012-05-26',
               '2012-05-27', '2012-05-28', '2012-05-29', '2012-05-30',
               '2012-05-31', '2012-06-01'],
              dtype='datetime64[ns]', freq='D')
```

```
In [76]: pd.date_range(start='2012-04-01', periods=20)
Out[76]:
DatetimeIndex(['2012-04-01', '2012-04-02', '2012-04-03', '2012-04-04',
               '2012-04-05', '2012-04-06', '2012-04-07', '2012-04-08',
               '2012-04-09', '2012-04-10', '2012-04-11', '2012-04-12',
               '2012-04-13', '2012-04-14', '2012-04-15', '2012-04-16',
               '2012-04-17', '2012-04-18', '2012-04-19', '2012-04-20'],
              dtype='datetime64[ns]', freq='D')

In [77]: pd.date_range(end='2012-06-01', periods=20)
Out[77]:
DatetimeIndex(['2012-05-13', '2012-05-14', '2012-05-15', '2012-05-16',
               '2012-05-17', '2012-05-18', '2012-05-19', '2012-05-20',
               '2012-05-21', '2012-05-22', '2012-05-23', '2012-05-24',
               '2012-05-25', '2012-05-26', '2012-05-27', '2012-05-28',
               '2012-05-29', '2012-05-30', '2012-05-31', '2012-06-01'],
              dtype='datetime64[ns]', freq='D')

In [78]: pd.date_range('2000-01-01', '2000-12-01', freq='BM')
Out[78]:
DatetimeIndex(['2000-01-31', '2000-02-29', '2000-03-31', '2000-04-28',
               '2000-05-31', '2000-06-30', '2000-07-31', '2000-08-31',
               '2000-09-29', '2000-10-31', '2000-11-30'],
              dtype='datetime64[ns]', freq='BM')
```

Alias	Offset type	Description
D	Day	Calendar daily
B	BusinessDay	Business daily
H	Hour	Hourly
T or min	Minute	Minutely
S	Second	Secondly
L or ms	Milli	Millisecond (1/1,000 of 1 second)
U	Micro	Microsecond (1/1,000,000 of 1 second)
M	MonthEnd	Last calendar day of month
BM	BusinessMonthEnd	Last business day (weekday) of month
MS	MonthBegin	First calendar day of month
BMS	BusinessMonthBegin	First weekday of month
W-MON, W-TUE, ...	Week	Weekly on given day of week (MON, TUE, WED, THU, FRI, SAT, or SUN)
WOM-1MON, WOM-2MON, ...	WeekOfMonth	Generate weekly dates in the first, second, third, or fourth week of the month (e.g., WOM-3FRI for the third Friday of each month)
Q-JAN, Q-FEB, ...	QuarterEnd	Quarterly dates anchored on last calendar day of each month, for year ending in indicated month (JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, or DEC)
BQ-JAN, BQ-FEB, ...	BusinessQuarterEnd	Quarterly dates anchored on last weekday day of each month, for year ending in indicated month
QS-JAN, QS-FEB, ...	QuarterBegin	Quarterly dates anchored on first calendar day of each month, for year ending in indicated month
BQS-JAN, BQS-FEB, ...	BusinessQuarterBegin	Quarterly dates anchored on first weekday day of each month, for year ending in indicated month
A-JAN, A-FEB, ...	YearEnd	Annual dates anchored on last calendar day of given month (JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, or DEC)
BA-JAN, BA-FEB, ...	BusinessYearEnd	Annual dates anchored on last weekday of given month
AS-JAN, AS-FEB, ...	YearBegin	Annual dates anchored on first day of given month
BAS-JAN, BAS-FEB, ...	BusinessYearBegin	Annual dates anchored on first weekday of given month

```
In [79]: pd.date_range('2012-05-02 12:56:31', periods=5)
Out[79]:
DatetimeIndex(['2012-05-02 12:56:31', '2012-05-03 12:56:31',
               '2012-05-04 12:56:31', '2012-05-05 12:56:31',
               '2012-05-06 12:56:31'],
              dtype='datetime64[ns]', freq='D')

In [80]: pd.date_range('2012-05-02 12:56:31', periods=5, normalize=True)
Out[80]:
DatetimeIndex(['2012-05-02', '2012-05-03', '2012-05-04', '2012-05-05',
               '2012-05-06'],
              dtype='datetime64[ns]', freq='D')
```

Frequencies and Date Offsets

```
In [81]: from pandas.tseries.offsets import Hour, Minute

In [82]: hour = Hour()

In [83]: hour
Out[83]: <Hour>

In [84]: four_hours = Hour(4)

In [85]: four_hours
Out[85]: <4 * Hours>
```

```
In [86]: pd.date_range('2000-01-01', '2000-01-03 23:59', freq='4h')
Out[86]:
DatetimeIndex(['2000-01-01 00:00:00', '2000-01-01 04:00:00',
               '2000-01-01 08:00:00', '2000-01-01 12:00:00',
               '2000-01-01 16:00:00', '2000-01-01 20:00:00',
               '2000-01-02 00:00:00', '2000-01-02 04:00:00',
               '2000-01-02 08:00:00', '2000-01-02 12:00:00',
               '2000-01-02 16:00:00', '2000-01-02 20:00:00',
               '2000-01-03 00:00:00', '2000-01-03 04:00:00',
               '2000-01-03 08:00:00', '2000-01-03 12:00:00',
               '2000-01-03 16:00:00', '2000-01-03 20:00:00'],
              dtype='datetime64[ns]', freq='4H')
```

```
In [87]: Hour(2) + Minute(30)
Out[87]: <150 * Minutes>

In [88]: pd.date_range('2000-01-01', periods=10, freq='1h30min')
Out[88]:
DatetimeIndex(['2000-01-01 00:00:00', '2000-01-01 01:30:00',
               '2000-01-01 03:00:00', '2000-01-01 04:30:00',
               '2000-01-01 06:00:00', '2000-01-01 07:30:00',
               '2000-01-01 09:00:00', '2000-01-01 10:30:00',
               '2000-01-01 12:00:00', '2000-01-01 13:30:00'],
              dtype='datetime64[ns]', freq='90T')

In [89]: rng = pd.date_range('2012-01-01', '2012-09-01', freq='WOM-3FRI')

In [90]: list(rng)
Out[90]:
[Timestamp('2012-01-20 00:00:00', freq='WOM-3FRI'),
 Timestamp('2012-02-17 00:00:00', freq='WOM-3FRI'),
 Timestamp('2012-03-16 00:00:00', freq='WOM-3FRI'),
 Timestamp('2012-04-20 00:00:00', freq='WOM-3FRI'),
 Timestamp('2012-05-18 00:00:00', freq='WOM-3FRI'),
 Timestamp('2012-06-15 00:00:00', freq='WOM-3FRI'),
 Timestamp('2012-07-20 00:00:00', freq='WOM-3FRI'),
 Timestamp('2012-08-17 00:00:00', freq='WOM-3FRI')]
```

```
In [91]: ts = pd.Series(np.random.randn(4),
   ....:                 index=pd.date_range('1/1/2000', periods=4, freq='M'))

In [92]: ts
Out[92]:
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 [93]: ts.shift(2)
Out[93]:
2000-01-31         NaN
2000-02-29         NaN
2000-03-31   -0.066748
2000-04-30    0.838639
Freq: M, dtype: float64

In [94]: ts.shift(-2)
Out[94]:
2000-01-31   -0.117388
2000-02-29   -0.517795
2000-03-31         NaN
2000-04-30         NaN
Freq: M, dtype: float64
```

```
ts / ts.shift(1) - 1
```

```
In [95]: ts.shift(2, freq='M')
Out[95]:
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 [96]: ts.shift(3, freq='D')
Out[96]:
2000-02-03   -0.066748
2000-03-03    0.838639
2000-04-03   -0.117388
2000-05-03   -0.517795
dtype: float64
```

```
In [97]: ts.shift(1, freq='90T')
Out[97]:
2000-01-31 01:30:00   -0.066748
2000-02-29 01:30:00    0.838639
2000-03-31 01:30:00   -0.117388
2000-04-30 01:30:00   -0.517795
Freq: M, dtype: float64
```

Shifting dates with offsets

```
In [98]: from pandas.tseries.offsets import Day, MonthEnd

In [99]: now = datetime(2011, 11, 17)

In [100]: now + 3 * Day()
Out[100]: Timestamp('2011-11-20 00:00:00')

In [101]: now + MonthEnd()
Out[101]: Timestamp('2011-11-30 00:00:00')

In [102]: now + MonthEnd(2)
Out[102]: Timestamp('2011-12-31 00:00:00')

In [103]: offset = MonthEnd()

In [104]: offset.rollforward(now)
Out[104]: Timestamp('2011-11-30 00:00:00')

In [105]: offset.rollback(now)
Out[105]: Timestamp('2011-10-31 00:00:00')
```