UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

Essential Data Skills for Business Analytics

Lecture 11 (04/10, 04/12): Pandas

Decision, Operations & Information Technologies
Robert H. Smith School of Business
Spring, 2017

Pandas

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- **Pan**el **Da**ta **S**ystem
- Pandas is an open source, BSD-licensed library
- High-performance, easy-to-use data structures and data analysis tools
- Built for the Python programming language

- Key components
  - ❑Series
  - ❑DataFrame

Import module

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
# Import all libraries needed

# General syntax to import specific functions in a library:
##from (library) import (specific library function)
from pandas import DataFrame, read_csv

# General syntax to import a library but no functions:
##import (library) as (give the library a nickname/ alias)
import pandas as pd #this is how I usually import pandas
```

Data structures: Series

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- One-dimensional array like object containing data and labels (or index)
- Lots of ways to build a Series

```
Master:code kpzhang$ python
Python 2.7.11 (v2.7.11:6d1b6a68f775, Dec  5 2015, 12:54:16)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas as pd
>>> s = pd.Series(['a','b','c','d','e','f'])
>>> s
0    a
1    b
2    c
3    d
4    e
5    f
dtype: object
>>>
```

Data structures: Series

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
>>> sdata = {'a':100,'b':150,'c':200}
>>> s = pd.Series(sdata)
>>> s
a    100
b    150
c    200
dtype: int64
>>>
```

Series – working with index

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- A Series index can be specified
- Single values can be selected by index
- Multiple values can be selected with multiple indexes

```
>>> s = pd.Series([2,4,6,8], index=['a','b','c','d'])
>>> s
a    2
b    4
c    6
d    8
dtype: int64
>>> s['c']
6
>>> s[['c','d']]
c    6
d    8
dtype: int64
>>>
```

Series – working with index

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Think of a Series as a fixed-length order dictionary
- However, unlike dictionary, index items don't have to be unique

```
>>> s2 = pd.Series(range(4),index = ['a','b','a','b'])
>>> s2
a    0
b    1
a    2
b    3
dtype: int64
>>> s2['a']
a    0
a    2
dtype: int64
>>> s2['a'][1]
2
>>>
```

Series operations

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Filtering
- NumPy-type operations on data

```
>>> s2[s2>0]
b     1
a     2
b     3
dtype: int64
>>> s2>2
a    False
b    False
a    False
b     True
dtype: bool
>>> s2*2
a     0
b     2
a     4
b     6
dtype: int64
>>>
```

Data structures: DataFrame

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Spreadsheet-like data structure containing an order collection of columns
- Has both a row and column index
- Consider as dictionary of Series (with shared index)

Create data

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Creation with dict of equal-length lists

```
>>> data = {'state': ['FL', 'FL', 'GA', 'GA', 'GA'],
            'year':  [2010, 2011, 2008, 2010, 2011],
            'pop':   [18.8, 19.1, 9.7, 9.7, 9.8]}
>>> frame = pd.DataFrame(data)
>>> frame
    pop state  year
0  18.8    FL  2010
1  19.1    FL  2011
2   9.7    GA  2008
3   9.7    GA  2010
4   9.8    GA  2011
```

Create data

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Creation with dict of dicts

```
>>> pop_data = {'FL': {2010:18.8, 2011:19.1},
                'GA': {2008: 9.7, 2010: 9.7, 2011:9.8}}
>>> pop = pd.DataFrame(pop_data)
>>> pop
        FL   GA
2008   NaN  9.7
2010  18.8  9.7
2011  19.1  9.8
```

Create data

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
# The inital set of baby names and bith rates
>>> names = ['Bob','Jessica','Mary','John','Mel']
>>> births = [968, 155, 77, 578, 973]
>>> BabyDataSet = list(zip(names, births))
>>> BabyDataSet
[('Bob', 968), ('Jessica', 155), ('Mary', 77), ('John', 578), ('Mel', 973)]
```

Create data

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
# create a Pandas DataFrame object: df
>>> df = pd.DataFrame(data = BabyDataSet, columns=['Names', 'Births'])
>>> df
     Names  Births
0  Bob      968
1  Jessica  155
2  Mary     77
3  John     578
4  Mel      973
```

You can think of df holding the contents of the BabyDataSet in a format similar to a sql table or an excel spreadsheet.

Create data

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
# export the dataframe to a csv file
>>> df.to_csv('filename')
>>> df.to_csv('filename', index=False, header=False)
```

The file will be saved in the same location as your python file unless specified otherwise.

Loading data


```
# To pull in the csv file, we can use the pandas function: read_csv
>>> filepath = 'filename'
>>> df = pd.read_csv(filepath, header=None)
>>> df
```

	0	1
0	Bob	968
1	Jessica	155
2	Mary	77
3	John	578
4	Mel	973

Loading data

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

If we wanted to give the columns specific names, we would have to pass another paramter called names.

```
>>> df = pd.read_csv(filepath, names=['Names', 'Births'])
>>> df
     Names    Births
0    Bob      968
1    Jessica  155
2    Mary     77
3    John     578
4    Mel      973
```

Prepare data

df.head() returns the first 5 records

df.head(n) returns the first n records

df.tail() returns the last 5 records

df.tail(n) returns the last n records

Prepare data

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
# check data type of the columns
>>> df.dtypes
Names     object
Births    int64
dtype: object

>>> df.Births.dtype
dtype('int64')
```

Prepare data

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
# to find all the unique records of the "Names" column
>>> df['Names'].unique()

# to obtain all descriptive statistics

>>> df['Names'].describe()
```

Analyze data

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
# To find the baby name with the highest birth rate
>>> sorted = df.sort_values(['Births'], ascending=False)
>>> sorted.head(1)
```

	Names	Births
4	Mel	973

```
>>> df['Births'].max()
```

Analyze data

df['Names'] - This is the entire list of baby names, the entire Names column

df['Births'] - This is the entire list of Births in the year 1880, the entire Births column

df['Births'].max() - This is the maximum value found in the Births column

[df['Births'] == df['Births'].max()] **IS EQUAL TO** [Find all of the records in the Births column where it is equal to 973]

df['Names'][df['Births'] == df['Births'].max()] **IS EQUAL TO** Select all of the records in the Names column WHERE [The Births column is equal to 973]

Column operations

```
>>> d = range(10)
>>> df = pd.DataFrame(d)
>>> df.columns = ['Rev']
>>> df['NewCol'] = 5
>>> df['NewCol'] = df['NewCol'] + 1
```

	Rev	NewCol
0	0	6
1	1	6
2	2	6
3	3	6
4	4	6
5	5	6
6	6	6
7	7	6
8	8	6
9	9	6

Column operations


```
>>> d = range(10)
>>> df = pd.DataFrame(d)
>>> df.columns = ['Rev']
>>> df['test'] = 3
>>> df['col'] = df['Rev']
>>> df
```

	Rev	test	col
0	0	3	0
1	1	3	1
2	2	3	2
3	3	3	3
4	4	3	4
5	5	3	5
6	6	3	6
7	7	3	7
8	8	3	8
9	9	3	9

Column operations

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
>>> d = range(10)
>>> df = pd.DataFrame(d)
>>> df.columns = ['Rev']
>>> df['test'] = 3
>>> df['col'] = df['Rev']
>>> i = ['a','b','c','d','e','f','g','h','i','j']
>>> df.index = i
```

	Rev	test	col
a	0	3	0
b	1	3	1
c	2	3	2
d	3	3	3
e	4	3	4
f	5	3	5
g	6	3	6
h	7	3	7
i	8	3	8
j	9	3	9

Column operations

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
>>> df.loc['a']
Rev   0
test  3
col   0
Name: a, dtype: int64
>>> df.loc['a' : 'd'] # both inclusive
```

	Rev	test	col
a	0	3	0
b	1	3	1
c	2	3	2
d	3	3	3

```
>>> df.iloc[0 : 3] # inclusive : exclusive
```

	Rev	test	col
a	0	3	0
b	1	3	1
c	2	3	2

Column operations

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
>>> df[ ['Rev', 'test'] ]

# df.ix[rows, columns]
>>> df.ix[0:3, 'Rev']

>>> df.ix[5 : , 'col']

>>> df.ix[:3, ['col', 'test']]
```

	Rev	test
a	0	3
b	1	3
c	2	3
d	3	3
e	4	3
f	5	3
g	6	3
h	7	3
i	8	3
j	9	3

	col	test
a	0	3
b	1	3
c	2	3

Groupby function

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
# Our small data set
d = {'one':[1,1,1,1,1],
     'two':[2,2,2,2,2],
     'letter':['a','a','b','b','c']}

# Create dataframe
df = pd.DataFrame(d)
df
```

	letter	one	two
0	a	1	2
1	a	1	2
2	b	1	2
3	b	1	2
4	c	1	2

```
# Create group object
one = df.groupby('letter')

# Apply sum function
one.sum()
```

	one	two
letter		
a	2	4
b	2	4
c	1	2

Groupby function


```
letterone = df.groupby(['letter','one']).sum()
letterone
```

		two
letter	one	
a	1	4
b	1	4
c	1	2

```
letterone = df.groupby(['letter','one'], as_index=False).sum()
letterone
```

	letter	one	two
0	a	1	4
1	b	1	4
2	c	1	2

Pandas missing data

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
>>> import numpy as np
>>>df = pd.DataFrame(np.random.randn(5, 3)
    , index=['a', 'c', 'e', 'f', 'h'],
columns=['one', 'two', 'three'])
>>>df.loc['a','two'] = np.nan
        one        two      three
a -1.192838        NaN -0.337037
c  0.110718 -0.016733 -0.137009
e  0.153456  0.266369 -0.064127
f  1.709607 -0.424790 -0.792061
h -1.076740 -0.872088 -0.436127
```

Pandas missing data

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
>>>df.isnull()
      one    two  three
a   False   True  False
c   False  False  False
e   False  False  False
f   False  False  False
h   False  False  False
```

Pandas missing data

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
>>>df.fillna(0)
        one       two     three
a -1.192838  0.000000 -0.337037
c  0.110718 -0.016733 -0.137009
e  0.153456  0.266369 -0.064127
f  1.709607 -0.424790 -0.792061
h -1.076740 -0.872088 -0.436127
```

Pandas query data

Use the query method where you can embed boolean expressions on columns within quotes

```
>>>df.query('one > 0')
          one       two     three
c   0.110718 -0.016733 -0.137009
e   0.153456  0.266369 -0.064127
f   1.709607 -0.424790 -0.792061
>>>df.query('one > 0 & two > 0')
          one       two     three
e   0.153456  0.266369 -0.064127
```

Pandas query data

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

You can apply any function to the columns in a dataframe

```
>>>df.apply(lambda x: x.max() - x.min())
one      2.902445
two      1.138457
three    0.727934
```

You can apply any function to the element wise data in a dataframe

```
>>>df.applymap(np.sqrt)
          one       two   three
a         NaN       NaN     NaN
c    0.332742       NaN     NaN
e    0.391735  0.516109     NaN
f    1.307520       NaN     NaN
h         NaN       NaN     NaN
```

Visualization

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- http://pandas.pydata.org/pandas-docs/stable/visualization.html