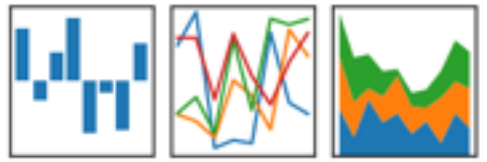
Columbia Data Science Society

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



Python Pandas Library for Data Science

How to write simple and efficient code using Pandas DataFrames

Columbia Data Science Society

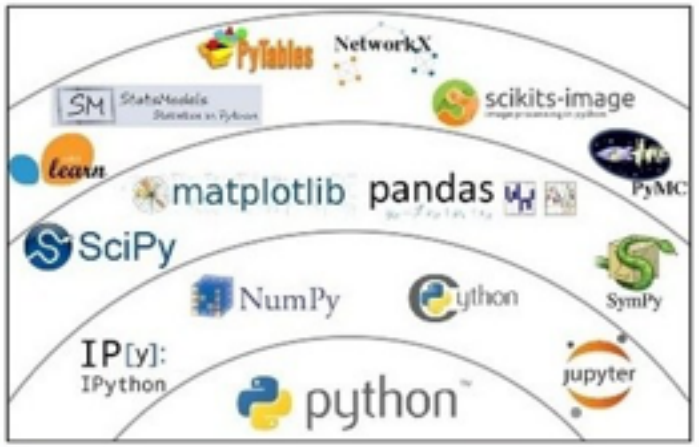
pandas

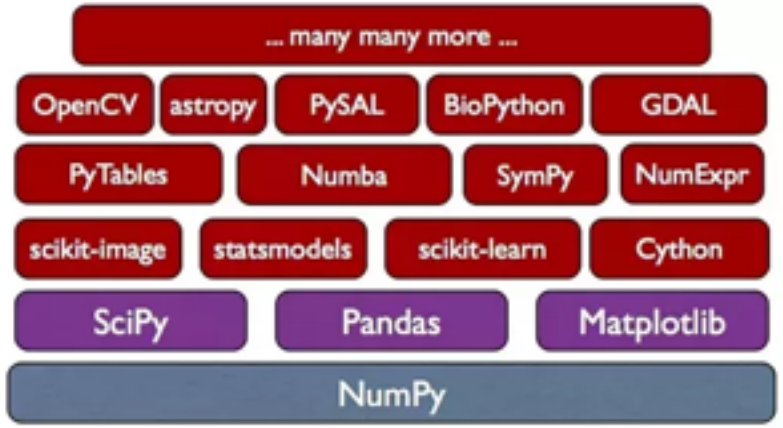
Contents

- DataFrames and Series
- Setting up a DataFrame
- File I/O
- Groupby
- Joins and Sets
- Cleaning Data
- Functions and Methods
- Strategy

DataFrames and Series

- A DataFrame is a two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). Arithmetic operations align on both row and column labels. **Can be thought of as a dict-like container for Series objects.** The primary pandas data structure
- A Series is a single column of a DataFrame
- Pandas objects are based on/built on top of NumPy arrays and add functionality for tables and time-series.
  - Joins, groupby, NaN-friendly functionality, data alignment





Columbia Data Science Society

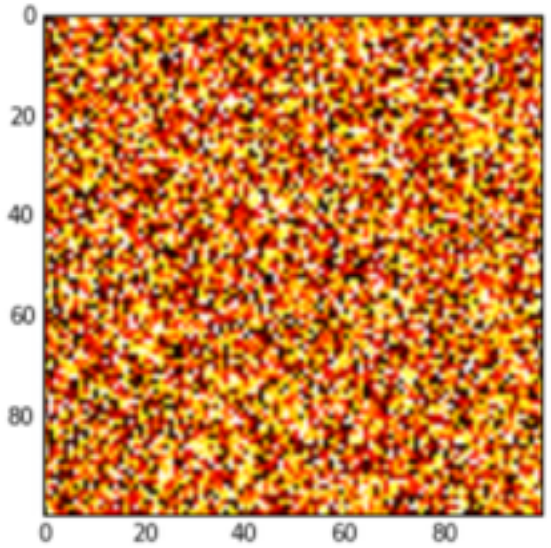
pandas

NumPy Arrays

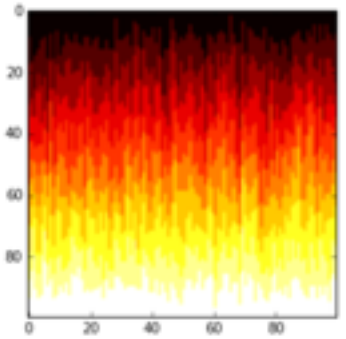
```
In [231]: Identity = np.identity(3)
          m1 = np.array([[4,2,3],
                         [3,2,1],
                         [1,3,3]])
          print(np.dot(Identity,m1))
```

```
[[ 4.  2.  3.]
 [ 3.  2.  1.]
 [ 1.  3.  3.]]
```

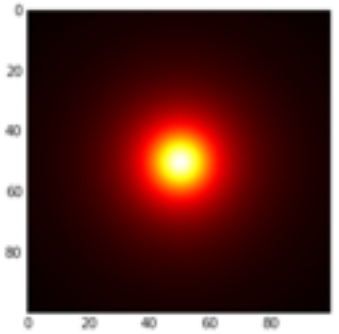
```
In [330]: heat_map = np.random.randint(10,size=[100,100])
          plt.imshow(heat_map,cmap='hot')
          plt.show()
```




```
In [331]: heat_map2 = np.sort(heat_map,axis=0)
          plt.imshow(heat_map2,cmap='hot')
          plt.show()
```




```
In [460]: import math
          heat_map3 = np.zeros((100,100))
          bulbx,bulby=50,50
          heat_map3[bulbx,bulby]=200
          for i in range(0,heat_map.shape[0]):
              for j in range(0,heat_map.shape[1]):
                  heat_map3[i,j]=(200/(4 * math.pi * math.sqrt((bulbx-i)**2 + (bulby-j)**2 + (10)**2)**2))
          plt.imshow(heat_map3,cmap='hot')
          plt.show()
```



Setting up a DataFrame

```
import pandas as pd
import numpy as np
```

```
In [307]: series = pd.Series(np.random.randn(1000))
          series.head()

Out[307]: 0    0.887445
          1    0.989532
          2    0.180761
          3    0.915032
          4    2.314938
          dtype: float64
```

```
In [308]: series.index = pd.date_range('20151020', periods=1000)
          series.head()

Out[308]: 2015-10-20    0.887445
          2015-10-21    0.989532
          2015-10-22    0.180761
          2015-10-23    0.915032
          2015-10-24    2.314938
          Freq: D, dtype: float64
```

Columbia Data Science Society

pandas

Setting up a DataFrame

```
In [232]: df1 = pd.DataFrame(columns=['country', 'capital', 'population'])
          df1
```

Out[232]:

	country	capital	population

```
In [87]: value1 = {'country':'Germany','capital':'Berlin', 'population':82700000}
         value2 = {'country':'USA','capital':'Washington D.C', 'population':323000000}
```

```
In [233]: df1=df1.append([value1,value2])
          df1
```

Out[233]:

	country	capital	population
0	Germany	Berlin	82700000
1	USA	Washington D.C	323000000

Columbia Data Science Society

pandas

Setting up a DataFrame

```
In [196]: array = np.random.randint(0,10,18)
          array.shape=(6,3)
          print(array)
```

```
[[9 1 1]
 [7 1 2]
 [1 7 2]
 [5 7 5]
 [0 9 7]
 [7 8 0]]
```

```
In [9]: df2 = pd.DataFrame(array, columns = ['A','B','C'], index = dates)
```

```
In [10]: df2
```

Out[10]:

	A	B	C
2017-10-20	4	7	1
2017-10-21	9	3	6
2017-10-22	8	3	8
2017-10-23	7	4	2
2017-10-24	1	4	4
2017-10-25	6	7	3

Columbia Data Science Society

pandas

File I/O

```
In [11]: df_titanic = pd.read_csv('titanic.csv')
```

```
In [13]: df_titanic.to_csv('titanic2.csv')
```

```
In [375]: df_titanic
```

Out[375]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	2	NaN	St Louis, MO
1	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON
2	1	0	Allison, Miss. Helen Loraine	female	2.0000	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON
3	1	0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1	2	113781	151.5500	C22 C26	S	NaN	135	Montreal, PQ / Chesterville,

Columbia Data Science Society

pandas

Groupby

```
In [377]: #show all members of a group
          g1 = df_titanic.groupby('cabin')
          for i,j in g1:
              print(i)
              print(j.name)
```

```
Name: name, dtype: object
A31
31    Blank, Mr. Henry
Name: name, dtype: object
A32
241    Rood, Mr. Hugh Roscoe
Name: name, dtype: object
A34
93                    Dodge, Dr. Washington
94                Dodge, Master. Washington
95    Dodge, Mrs. Washington (Ruth Vidaver)
Name: name, dtype: object
A36
7    Andrews, Mr. Thomas Jr
Name: name, dtype: object
A5
135    Goldschmidt, Mr. George B
Name: name, dtype: object
A6
```

Groupby

```
In [378]: #show all members of a group
          g1 = df_titanic.groupby('survived')
          for i,j in g1:
              print(i)
              print(j.name)
```

```
0.0
2                             Allison, Miss. Helen Loraine
3                   Allison, Mr. Hudson Joshua Creighton
4          Allison, Mrs. Hudson J C (Bessie Waldo Daniels)
7                               Andrews, Mr. Thomas Jr
9                              Artagaveytia, Mr. Ramon
10                             Astor, Col. John Jacob
15                                Baumann, Mr. John D
16                            Baxter, Mr. Quigg Edmond
19                                Beattie, Mr. Thomson
25                                Birnbaum, Mr. Jakob
30                           Blackwell, Mr. Stephen Weart
34                               Borebank, Mr. John James
38                              Brady, Mr. John Bertram
39                                   Brandeis, Mr. Emil
40                             Brewe, Dr. Arthur Jackson
45                  Butt, Major. Archibald Willingham
46                               Cairns, Mr. Alexander
51                             Carlsson, Mr. Frans Olof
```

Columbia Data Science Society

pandas

Groupby

```
In [405]: #group by combined with a method
          for i, j in g1:
              print(i)
              print(j.age.mean())
```

```
0.0
30.54536882067851
1.0
28.918228103044495
```

```
In [380]: df_titanic['fare_range']=pd.cut(df_titanic.fare, bins=[-1,0,10,20,50,100,int(df_titanic.fare.max()+1)])

In [381]: g2=df_titanic.groupby(df_titanic.fare_range)

In [382]: for i,j in g2:
              print(i)
              print(j.name.count())
```

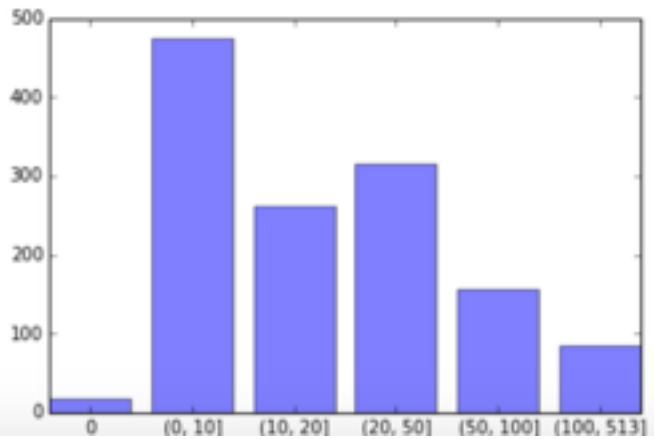
```
(-1, 0]
17
(0, 10]
474
(10, 20]
261
(20, 50]
316
(50, 100]
156
(100, 513]
84
```

Columbia Data Science Society

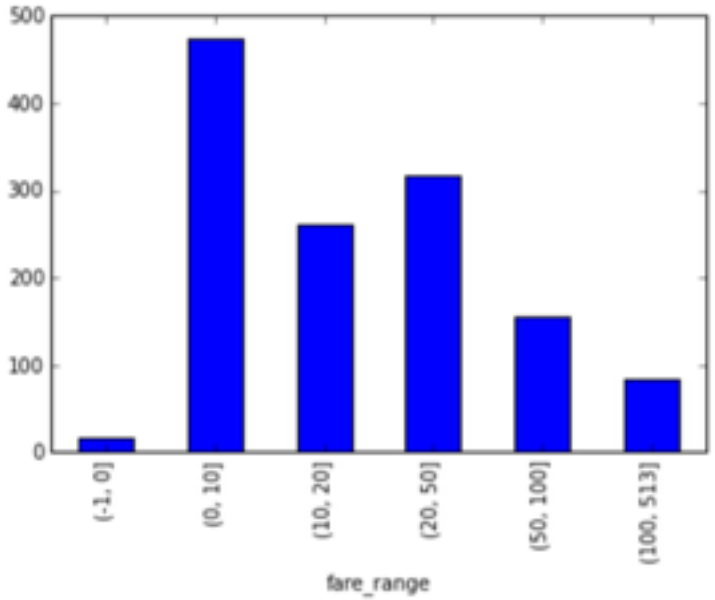
pandas

Groupby

```
In [383]: #the iterative, inefficient way
          l1=[]
          l2=[]
          for i,j in g2:
              l1.append(i)
              l2.append(j.name.count())

In [384]: %matplotlib inline
          import matplotlib.pyplot as plt

In [385]: l1[0]=0

In [386]: objects =l1
          y_pos = np.arange(len(l1))
          plt.xticks(y_pos,objects)
          plt.bar(y_pos,l2,align='center', alpha=0.5)
          plt.xlabel('Fare Range')

          plt.show()
```




```
In [387]: #the pandas way
          g2['name'].count().plot(kind='bar')

Out[387]: <matplotlib.axes._subplots.AxesSubplot at 0x11a3029b0>
```



Columbia Data Science Society

pandas

Joins

Joins/ Merges

Whats the difference?

1. Merge does an inner join by default, while join does a left join.
2. Join by default joins on the index of the two dataframes.

```
In [409]: #Make sure you have an index/ID Column to ensure a correct result
          df_titanic.index.rename('ID', inplace=True)

In [410]: passengers = df_titanic[['name','sex', 'age']]

In [411]: found_bodies = df_titanic[['body']]
```

```
In [391]: passengers.join(found_bodies)
```

Out[391]:

	name	sex	age	body
ID				
0	Allen, Miss. Elisabeth Walton	female	29.0000	NaN
1	Allison, Master. Hudson Trevor	male	0.9167	NaN
2	Allison, Miss. Helen Loraine	female	2.0000	NaN

```
In [392]: pd.merge(passengers,found_bodies, how='left', left_index=True, right_index=True)
```

Out[392]:

	name	sex	age	body
ID				
0	Allen, Miss. Elisabeth Walton	female	29.0000	NaN
1	Allison, Master. Hudson Trevor	male	0.9167	NaN
2	Allison, Miss. Helen Loraine	female	2.0000	NaN

Columbia Data Science Society

pandas

Sets

```
In [164]: df_young = df_titanic.loc[df_titanic.age <=20]
          df_old = df_titanic.loc[df_titanic.age >=20]
```

```
In [170]: #Union
          df_all = pd.concat([df_young, df_old])
          df_all
```

339	2	1	Becker, Master. Richard F	male	1.0000	2	1	230136	39.0000	F4	S	11	NaN	Guntur, India / Benton Harbour, MI	(20, 50]
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1258	3	1	Touma, Mrs. Darwis (Hanne Youssef Razi)	female	29.0000	0	2	2650	15.2458	NaN	C	C	NaN	NaN	(10, 20]
1259	3	0	Turcin, Mr. Stjepan	male	36.0000	0	0	349247	7.8958	NaN	S	NaN	NaN	NaN	(0, 10]
1261	3	1	Turkula, Mrs. (Hedwig)	female	63.0000	0	0	4134	9.5875	NaN	S	15	NaN	NaN	(0, 10]
1264	3	0	van Billiard, Mr. Austin Blyler	male	40.5000	0	2	A/5. 851	14.5000	NaN	S	NaN	255	NaN	(10, 20]

Columbia Data Science Society

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

Sets

```
In [195]: #alternative using series union
          df_titanic.loc[df_young.index.union(df_old.index)]
```

1300	3	1	Antoni (Selini Alexander)	female	15.0000	1	0	2659	14.4542	NaN	C	NaN	NaN	NaN	(10, 20]
1301	3	0	Youseff, Mr. Gerious	male	45.5000	0	0	2628	7.2250	NaN	C	NaN	312	NaN	(0, 10]
1304	3	0	Zabour, Miss. Hileni	female	14.5000	1	0	2665	14.4542	NaN	C	NaN	328	NaN	(10, 20]
1306	3	0	Zakarian, Mr. Mapriededer	male	26.5000	0	0	2656	7.2250	NaN	C	NaN	304	NaN	(0, 10]
1307	3	0	Zakarian, Mr. Ortin	male	27.0000	0	0	2670	7.2250	NaN	C	NaN	NaN	NaN	(0, 10]
1308	3	0	Zimmerman, Mr. Leo	male	29.0000	0	0	315082	7.8750	NaN	S	NaN	NaN	NaN	(0, 10]

1046 rows × 15 columns

Columbia Data Science Society

pandas

Sets

```
In [171]: df_young2 = df_titanic.loc[df_titanic.age <=30]
          df_old2 = df_titanic.loc[df_titanic.age >=20]

In [172]: #intersection
          df_inter = pd.merge(df_young2, df_old2, how='inner', left_index=True, right_index=True)

In [173]: df_inter
```

Out[173]:

	pclass_x	survived_x	name_x	sex_x	age_x	sibsp_x	parch_x	ticket_x	fare_x	cabin_x	...	sibsp_y	parch_y	ticket_y	fare_y
ID															
0	1	1	Allen, Miss. Elisabeth Walton	female	29.0	0	0	24160	211.3375	B5	...	0	0	24160	211.3375
3	1	0	Allison, Mr. Hudson Joshua Creighton	male	30.0	1	2	113781	151.5500	C22 C26	...	1	2	113781	151.5500
4	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0	1	2	113781	151.5500	C22 C26	...	1	2	113781	151.5500

Sets

```
In [213]: #alternative using series intersection
          df_titanic.loc[df_young2.index.intersection(df_old2.index)]
```

Out[213]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
ID														
0	1	1	Allen, Miss. Elisabeth Walton	female	29.0	0	0	24160	211.3375	B5	S	2	NaN	St Louis, MO
3	1	0	Allison, Mr. Hudson Joshua Creighton	male	30.0	1	2	113781	151.5500	C22 C26	S	NaN	135	Montreal, PQ / Chesterville, ON
4	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON
12	1	1	Aubart, Mme. Leontine Pauline	female	24.0	0	0	PC 17477	69.3000	B35	C	9	NaN	Paris, France

Columbia Data Science Society

pandas

Cleaning Data

```
In [ ]: del df_titanic['fare_range']
        df_titanic
```

```
In [301]: from sklearn.preprocessing import Imputer
          s = [[1,np.NaN,1],
               [np.NaN,3,5],
               [8,9,None]]
          imp2 = Imputer(missing_values='NaN',strategy='mean',axis=0,verbose=0,copy=True)
          imp2.fit([[1,0,1],
                    [8,3,5],
                    [8,9,3]])
          print(imp2.transform(s))
```

```
[[ 1.          4.          1.        ]
 [ 5.66666667  3.          5.        ]
 [ 8.          9.          3.        ]]
```

```
In [402]: df2 = pd.DataFrame({'A':[1,np.NaN,5], 'B':[4,5,np.NaN]})
          df2.fillna(df2.mean())
```

Out[402]:

	A	B
0	1	4.0
1	3	5.0
2	5	4.5

Columbia Data Science Society

pandas

Cleaning Data

```
In [424]: df3 = pd.DataFrame({'A':[1,np.NaN,5,np.NaN,np.NaN,np.NaN], 'B':[4,5,np.NaN,5,2,1]})
          df3
```

Out[424]:

	A	B
0	1	4
1	NaN	5
2	5	NaN
3	NaN	5
4	NaN	2
5	NaN	1

```
In [425]: df3.dropna()
```

Out[425]:

	A	B
0	1	4

```
In [428]: df3.dropna(axis=1, thresh=3)
```

Out[428]:

	B
0	4
1	5
2	NaN
3	5
4	2
5	1

Columbia Data Science Society

pandas

Functions and Methods

Functions and Methods

```
In [403]: g4=df_titanic.groupby('boat')['survived']
          (g4.sum()/g4.count()).round(2)
```

```
Out[403]: boat
          1           1.00
          10          1.00
          11          1.00
          12          0.95
          13          1.00
          13 15       1.00
          13 15 B     1.00
          14          0.97
          15          1.00
          15 16       1.00
          16          1.00
          2           1.00
          3           1.00
          4           1.00
          5           1.00
          5 7         1.00
          5 9         1.00
          6           1.00
          7           1.00
          8           1.00
          8 10        1.00
          9           1.00
          A           0.64
          B           0.89
          C           0.97
          C D         1.00
```

```
In [404]: df_titanic.age.apply(lambda x: x.round(0)).head()
```

```
Out[404]: 0    29
          1     1
          2     2
          3    30
          4    25
          Name: age, dtype: float64
```

```
In [431]: df_titanic.age.round(0)
```

```
Out[431]: ID
          0        29
          1         1
          2         2
          3        30
          4        25
          5        48
          6        63
          7        39
          8        53
          9        71
          10       47
          11       18
          12       24
          13       26
          14       80
```

Factorize

```
In [449]: df_titanic['cabin_factorized']=pd.factorize(df_titanic.cabin)[0]
          df_titanic
```

Out[449]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest	cabin_factorize
ID															
0	1	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	2	NaN	St Louis, MO	0
1	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON	1

Columbia Data Science Society

pandas

Scaling

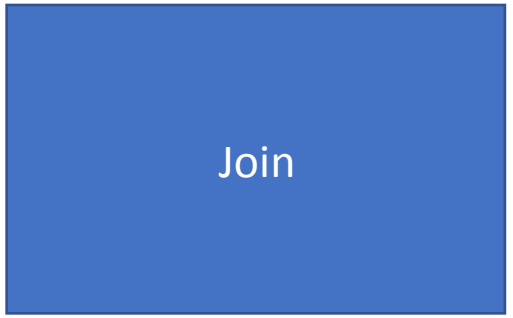
In [456]:
```
from sklearn.preprocessing import StandardScaler
df_titanic.dropna(subset =['age'],inplace=True)
scaler = StandardScaler()
df_titanic['Age_Scaled']=scaler.fit_transform(df_titanic['age'])
```

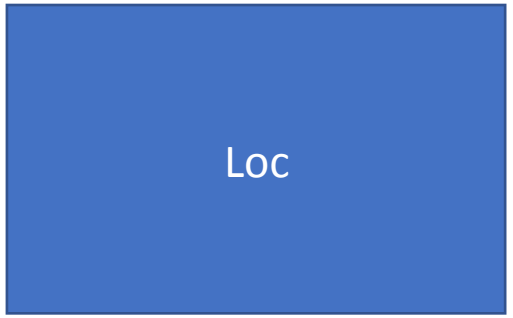
Columbia Data Science Society

pandas

Strategy

Use intermediary steps





Columbia Data Science Society

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

Strategy: Efficient Code

Speed

Power

User-friendliness

Strategy: Efficient Code

- Don’t try to write code that is consistent with the Python programs you are used to.
  - Don’t use Loops!
  - General rule: if your Pandas program looks like typical Python code you are most likely doing something wrong!
  - Take advantage of Pandas´ built-in optimizations instead.
  - Use Vectorization
- Don’t write complicated code for a method that most likely already exists in the library
  - Always check http://pandas.pydata.org/pandas-docs/stable/ or stackoverflow
- Check if you can convert your DataFrame to a NumPy Array: This can significantly increase speed.
  - This can be as easy as using df[‘column’].values instead of df[‘column’]

Methodology	Average single run time	Marginal performance improvement
Crude looping	645 ms	
Looping with iterrows()	166 ms	3.9x
Looping with apply()	90.6 ms	1.8x
Vectorization with Pandas series	1.62 ms	55.9x
Vectorization with NumPy arrays	0.37 ms	4.4x