TACC
TEXAS ADVANCED COMPUTING CENTER
WWW.TACC.UTEXAS.EDU

TEXAS
The University of Texas at Austin

Pandas 101

S. Charlie Dey, Director of Training and Professional Development

Pandas, What is it?

A software library written for the Python for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series

Pandas, The DataFrame

The primary pandas data structure.
Two-dimensional size-mutable, 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.

Pandas, First Steps

Let's create a simple data set, and see what Pandas can do.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Pandas, First Steps

Let's create a simple data set, and see what Pandas can do.

```
s = pd.Series([1,3,5,np.nan,6,8])
s
```

Pandas, First Steps

Let's create a simple data set, and see what Pandas can do.

```
dates = pd.date_range('20180101', periods=6)
dates
```

Pandas, First Steps

Let's create a simple data set, and see what Pandas can do.

```
df = pd.DataFrame(np.random.randn(6,4),
 index=dates, columns=list('ABCD'))
df
```

Pandas, First Steps

Let's create a simple data set, and see what Pandas can do.

```
df2 = pd.DataFrame({ 'A' : 1.,'B' :
 pd.Timestamp('20130102'),'C' :
 pd.Series(1,index=list(range(4)),dtype='float32'),'D' :
 np.array([3] * 4,dtype='int32'),'E' :
 pd.Categorical(["test","train","test","train"]),'F' :
 'foo' })


df2
```

Pandas, Viewing Data

Some common/useful functions

```
df.head()
df.tail(3)
df.index
df.columns
df.values
df.describe()
df.T
df.sort_index(axis=1, ascending=False)
df.sort_values(by='B')
```

Pandas, Selecting Data by Label

Some common/useful functions

```
df['A']
df[0:3]
df['20130102':'20130104']
df.loc[dates[0]]
df.loc[:,['A','B']]
df.loc['20130102':'20130104',['A','B']]
df.loc['20130102',['A','B']]
df.loc[dates[0],'A']
```

Pandas, Selecting Data by Position

Some common/useful functions

```
df.iloc[3]
df.iloc[3:5,0:2]
df.iloc[[1,2,4],[0,2]]
df.iloc[1:3,:]
df.iloc[:,1:3]
df.iloc[1,1]
df.iat[1,1]
```

Pandas, CSV Files

manipulating CSV files.

```
ts = pd.Series(np.random.randn(1000), index=pd.date_range('1/1/2000',
 periods=1000))
ts = ts.cumsum()  ## cumulative sum


df = pd.DataFrame(np.random.randn(1000, 4), index=ts.index,columns=['A',
 'B', 'C', 'D'])
df = df.cumsum()

df.to_csv('foo.csv')
pd.read_csv('foo.csv')
```

Pandas, CSV Files

filtering data made easy… .query

```
df = pd.DataFrame(np.random.randn(1000, 4), index=ts.index,columns=['A',
'B', 'C', 'D'])


df = df.cumsum()


df.query('A > 10 & B > 10')
```

Pandas, CSV Files

just becayse… return of lambda

```
df = pd.DataFrame(np.random.randn(1000, 4), index=ts.index,columns=['A',
'B', 'C', 'D'])


df = df.cumsum()


df.loc[lambda df: df.B > 10]  ## What do you think this does?
```

Pandas, Joining

Dataset 1

```
raw_data = {
        'subject_id': ['1', '2', '3', '4', '5'],
        'first_name': ['Alex', 'Amy', 'Allen', 'Alice', 'Ayoung'],
        'last_name': ['Anderson', 'Ackerman', 'Ali', 'Aoni', 'Atiches']}
df_a = pd.DataFrame(raw_data, columns = ['subject_id', 'first_name',
'last_name'])

df_a
```

Pandas, Joining

Dataset 2

```
raw_data = {
        'subject_id': ['4', '5', '6', '7', '8'],
        'first_name': ['Billy', 'Brian', 'Bran', 'Bryce', 'Betty'],
        'last_name': ['Bonder', 'Black', 'Balwner', 'Brice', 'Btisan']}

df_b = pd.DataFrame(raw_data, columns = ['subject_id', 'first_name',
'last_name'])

df_b
```

Pandas, Joining

Dataset 3

```
raw_data = {
        'subject_id': ['1', '2', '3', '4', '5', '7', '8', '9', '10', '11'],
        'test_id': [51, 15, 15, 61, 16, 14, 15, 1, 61, 16]}

df_n = pd.DataFrame(raw_data, columns = ['subject_id','test_id'])

df_n
```

Pandas, Joining

Joining along rows

```
df_new = pd.concat([df_a, df_b])
df_new
```

Pandas, Joining

Joining along columns

```
d.concat([df_a, df_b], axis=1)
```

Pandas, Joining

Merging

```
pd.merge(df_new, df_n, on='subject_id')
```

Pandas, Joining

Merging, Outer Join

```
pd.merge(df_a, df_b, on='subject_id', how='outer')
```

Pandas, Joining

Merging, Inner Join

```
pd.merge(df_a, df_b, on='subject_id', how='inner')
```

Pandas, Joining

Merging, Right Join

```
pd.merge(df_a, df_b, on='subject_id', how='right')
```

Pandas, Joining

Merging, Left Join

```
pd.merge(df_a, df_b, on='subject_id', how='left')
```

Pandas, Summary of Features

Pandas allow for:

Boolean Indexing
Statistical Operations
Histogramming
Merging Data
SQL Style Joins
SQL Style Appends
SQL Style Grouping
Reshaping
Pivoting
and more!

Questions? Comments?