Intermediate Data Science

Pandas - Advanced Topics

Joanna Bieri DATA201

Important Information

- Email: joanna_bieri@redlands.edu
- Office Hours take place in Duke 209 – Office Hours Schedule
- Class Website
- Syllabus

Review of DATA 101

In completing the review of DATA 101 you have already started working with Pandas! You actually know a lot about how Pandas works from our last class. The goal for today is to dig a bit deeper into the type of data that is stored in a Pandas DataFrame and to see some of the main capabilities of Pandas.

Data Structures in Pandas

The fundamental data types in Pandas are:

- **Series**: a one-dimensional object containing a sequence of values of the same type and an associated array of data labels.
- **DataFrame:** a rectangular table of data that contains an ordered, named collection of columns each of which could have a different data type. It has both a row and column index and it can be though of as a dictionary of Series.

Series Basics

```
example_series.index
example_series.array
example_series[['a','b']]
'b' in example_series
example_series.isna()
example_series.notna()
exampe_series.dropna()
```

DataFrames - Basics

Most of the time we will be working with DataFrames! Dictionaries work really well to define new data frames. You can access information using the column labels/keys or the row indexes

DataFrames - Basics

	state	year	pop
0	Ohio	2000	1.5
1	Ohio	2001	1.7
2	Ohio	2002	3.6
3	Nevada	2001	2.4
4	Nevada	2002	2.9
5	Nevada	2003	3.2

```
df.keys()
df['year'] vs. df.year

columns = ['state','pop']
df[columns]

df.head()
```

DataFrames - Data Manipulation

```
del df['debt']

# What does this do?
df['eastern'] = df['state'] == 'Ohio'
```

DataFrames - Functionality

```
df = df.reindex(index=["a", "b", "c", "d"])
df = df.T

df.loc['Ohio']
df.iloc[0]

df = df.drop('Ohio')
df = df.drop(columns='d')
df = df.dropna()
```

DataFrames - Filtering and Masking

```
# What is the difference?
df>3
df[df>3]
df[df>3] = 0

mask = df['Texas']>3
df[mask]

df.loc['a',['Texas','California']]
```

DataFrames - Arithmetic

```
df1 = pd.DataFrame({"A": [1, 2]})

df1*2
1/df1
df1**2

df1+df1

df2 = pd.DataFrame({"B": [3, 4]})
df1+df2
```

DataFrames - Functions

Often we want to apply a special function to parts of our data frame. We can do this using functions!

```
np.abs(df)
np.sqrt(df)

def max_min(x):
    return x.max()-x.min()

df.apply(max_min)

df.apply(lambda x: x.max()-x.min())
```

DataFrames - Sorting and Grouping

```
df.sort_values(by='b', ascending=False)
df.sort_index()

groups = df.groupby('location')
for g in groups:
    print(g[0])
    display(g[1])
```

DataFrames - Quick Descriptive Statistics

```
# NAME SOME REDUCTION METHODS:
df.sum(axis='index')
df.sum(axis='columns')
df.sum(axis='columns',skipna=False)

df.describe()
df.corr()
df.cov()
```

DataFrames - Value Counts and Membership

```
df['a'].value_counts().sort_index()
df['a'].unique()
```

DataFrames - Other Methods

In your Jupyter Lab Notebook, try entering a DataFrame and then looking into what methods are available! Use the question mark to look into the documentation.