Pandas!

Pandas is a Python module for data analysis. The central feature of Pandas is a data frame object. Data frames are useful objects for analyzing and manipulating tabular data. Think of a spreadsheet on steriods! The R programming language features a build in data frame class. In fact, data frames are the fundamental thing to work with in R. Newer versions of Matlab have the `table` class, which implements some of the functionality available in the Pandas and R data frames.

From the Pandas website:

- A fast and efficient DataFrame object for data manipulation with integrated indexing;
- Tools for reading and writing data between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format;
- Intelligent data alignment and integrated handling of missing data: gain automatic label-based alignment in computations and easily manipulate messy data into an orderly form;
- Flexible reshaping and pivoting of data sets;
- Intelligent label-based slicing, fancy indexing, and subsetting of large data sets;
- Columns can be inserted and deleted from data structures for size mutability;
- Aggregating or transforming data with a powerful group by engine allowing split-apply-combine operations on data sets;
- High performance merging and joining of data sets;
- Hierarchical axis indexing provides an intuitive way of working with high- dimensional data in a lower-dimensional data structure;
- Time series-functionality: date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging. Even create domain-specific time offsets and join time series without losing data;
- Highly optimized for performance, with critical code paths written in Cython or C.
- Python with pandas is in use in a wide variety of academic and commercial domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, and more.

First we must import pandas. The conventional way to do so is shown below:

```
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
%matplotlib inline
%config InlineBackend.figure_format = 'svg'
```

Import data

Let's look to see what is in this directory with a "magic" Jupyter command:

```
%ls
```

Ahh, we have a `csv` file. This is very easy to open in Pandas. Pandas can also open many other data types.

See: http://pandas.pydata.org/pandas-docs/version/0.17.0/io.html

Specifically, it is very easy to load data from MS Excel files (`pd.read_excel()`).

We have some basketball data from: http://www.basketball-reference.com/

```
nba = pd.read_csv("nba-2018.csv")
```

Let's inspect the data with the `head` method:

```
nba.head(10)
```

Exercise:

Modify the call to the `head` method to show more or fewer rows.

```
# Show slice of data rows
nba[100:130]
```

The output above does not show all of the columns! The dots denote skipped columns. We can inspect the set of columns in a Pandas data fram by looking at the `columns` attribute.

```
nba.columns
```

Exercise

Write a loop to nicely print out the column headers.

```
for label in nba.columns:
    print("{} ".format(label), end = ' ')
```

Column glossary

In many data sets, a code is use for column headers. It is important to know about the data you are working with. Here is what the columns in this dataset mean:

Rk	
Player	
Pos	Position
Age	
Tm	Team
G	Games
GS	Games started
MP	Minutes played
FG	Field goals
FGA	Field goals attempted
FG%	Field goal percentage
3P	3 pt field goals
3PA	3 pt field goals attempted
3P	3 pt field goal percentace
2P	2 pt field goals
2PA	2 pt field goals attempted
2P	2 pt field goals percentage
eFG%	effective field goal percentage
FT	Free throws
FTA	Free throws attempted
FT%	Free throw percentage
ORB	Offensive rebounds
DRB	Defenseive rebounds
TRB	Total rebounds
AST	Assists
STL	Steals
BLK	Blocks
TOV	Turnovers
PF	Personal fouls
PTS	Total points

The `info()` method shows basic information about Pandas' `DataFrame` class:

```
nba.info()
```

Indexing

Operation	Syntax	Result
Select column	df[col]	Series
Select row by label	df.loc[label]	Series
Select row by integer location	df.iloc[loc]	Series
Slice rows	df[5:10]	DataFrame
Select rows by boolean vector	df[bool_vec]	DataFrame

- A *Series* object is a single column in the table
- When single row is selected it is retuned as a Series
- When multiple rows are selected they are returned as a DataFrame
- A *DataFrame* object is the table

Column selection

```
# extract a single column with column index name
nba['Player'].head()
```

```
nba['PTS'].head()
```

```
# select several columns by passing a sequence of column names
# (this returns a data frame)
nba[['Player','PTS']].head()
```

Row selection

Currently all row labels in this data set are integers, so row access via `nba.loc` and `nba.iloc` are equivalent.

```
nba.loc[100]
```

```
# select multiple rows with an integer index slice
nba[200:205]
```

```
# we can compute a boolean series with python inequality operators
(nba['PTS'] >= 20).head(10)
```

The result is true for every player who averaged more than 20 points per game.

```
# we can select all rows that pass a filter
nba[nba['PTS'] >= 20]
```

Or, we can use a filter to find a player statistcs by name:

```
nba[nba['Player'] == 'Stephen Curry']
```

Column modifications

An important statistics in basketball is assist to turnover ratio, yet it is not included in the original data set. We can create a column with such statistics like this:

```
# let's create a new column
nba['ATO'] = nba['AST']/nba['TOV']
nba.head(7)
```

As you can see, some of the ratios are infinity. This is for players who played few games, had no turnovers, but did assist. These are not relevant data points for this analysis, so we can filter them out like this:

```
# Pandas uses overloaded bitwise operators for logical operations
nba[(nba['ATO'] >= 3.0) & (nba['TOV'] > 0) & (nba['G'] > 40)]
```

Note that logical operations use overloaded bitwise operators `&`, `|` and '~', rather than Python's logical operators `and`, `or` and `not`.

```
# let's delete the column we just made
del nba['ATO']
nba.head()
```

Simple plotting

```
# histogram of single column
nba['PTS'].hist()
```

```
# scatter plot of two columns
plt.plot(nba['G'],nba['FG%'],'o',alpha=0.2)
plt.xlabel('games played')
plt.ylabel('field goal %')
```

```
# scatter matrix of multiple columns
pd.plotting.scatter_matrix(nba[['AST','FG','TRB','PF']], alpha=0.2)
plt.tight_layout()
```

Grouping and aggregation

In this data set, a player may have multiple rows based on teams they played for during the year. Let's get the total number of games played for all players.

```
nba_notot = nba[nba['Tm'] != 'TOT'] # Remove totals from team names
player_games = nba_notot[['Player','G']].groupby('Player').agg({'G':np.sum})
# creates dataframe with player names as string labels
# this is an example of "method chaining"
```

```
player_games.head()
```

```
player_games.loc['Dwyane Wade']
```

```
nba[nba['Player'] == 'Dwyane Wade']
```

Exercises

Let us find most travelled journeyman in the NBA.

What is the max number of teams any player has played for?

```
# Top ten journeymen
player_teams = nba[['Player','Tm']].groupby('Player').count()
player_teams.sort_values(by=['Tm'], ascending=False)[0:10]
```

What is the highest scoring team? What is the lowest scoring team?

```
nba['TPTS']=nba['PTS']*nba['G']/82.0 # Compute total points per game
fnba = nba[nba['Tm']!='TOT']         # Filter out totals for players on multiple teams
teams_points = fnba[['Tm','TPTS']].groupby('Tm').agg({'TPTS':np.sum})   # Sum for each team

# Top five scoring teams
best = teams_points.sort_values(by='TPTS', ascending=False)
best.head(5)

# Bottom five scoring teams
worst = teams_points.sort_values(by='TPTS')
worst.head(5)
```

What is highest scoring position? What is lowest scoring position? How many players in each position? What is average score per position?

```
# code here
```

More questions: * Is there a correlation between fouls and rebounds? What if we normalize by minutes played? * What player has the most points per minute played? * Plot a histogram of minutes played for players? * What team has the highest variance of points scored by individual players? * What team has the lowest variance of points scored by individual players?

References

- **Python for Data Analysis** by Wes McKinney (2012) http://proquest.safaribooksonline.com/book/programming/python/9781449323592/firstchapter
- Pandas online documentation: http://pandas.pydata.org/pandas-docs/stable