Pandas

Libraries Used:

Import pandas as pd (more)

- Used to compute statistical representations of our data as well as manipulate dataframes

Setup:

Extracting csv files from SQL

- To manipulate the data we have been using is recent weeks, we need to extract it from SQL
- Instructions are in intro slides and demo video
- Very important to save csv file in same folder as your Jupyter Notebook file

Pandas Functions:

.read_csv() (more)

- Reads the csv file into Pandas
- Make sure csv file is in same folder as Jupyter notebook file

.head(n) (more)

- Returns the first n rows in a dataframe

.info() (more)

- Prints info about the dataframe

.shape (more)

- Prints number tuple containing number of rows and columns (row, column)

pd.concat() (more)

- Adds the rows of one df to the end of another df
- Can specify axis and type of join

.drop_duplicates() (more)

- Eliminates duplicate rows within a df

.columns (more)

- Prints list of columns within a df

.rename() (more)

- Renames the column names
- Parameters include dictionary of column name changes in 'old name' : 'new name' format and inplace = True

.isnull() (more)

- Returns True for all null values in df and false for non-null values

.describe() (more)

- Prints numerical summaries including count, mean, standard deviation, min, median, max, and quartiles for each column containing integers

.value_counts() (more)

- Counts the number of times a specific string/int appears in a column

.corr() (more)

- Prints correlation between each pair of columns that contain integers only

.groupby() (more)

- Splits the df into groups based on the parameter inside the parentheses
- Typically followed by second function such as .get_group(), .mean(), or .count() because it will not return anything on its own

Extracting data frame columns (more)

- Use single brackets [] to extract a column from the df but double brackets [[]] to set the column as its own df

```
opponent_col = df['opponent']
opponent_col
```

```
opponent_df = df[['opponent']]
opponent_df.head()
```

Accessing data frame rows (more)

- Use the .loc[] function or the .iloc[] function
  - loc requires label indexing (put the label of the row in the brackets)
  - iloc requires integer indexing (put the index of the row in the brackets)

```
row1_loc = alt_df.loc['2011-09-10']
row1_loc
row1_iloc = alt_df.iloc[1]
row1_iloc
```

- To access multiple rows, you can slice the df

```
first_100 = df.iloc[:100]
first_100
```

Accessing specific values within a data frame (more)

- Access the row and then access the column
- This can easily be done in one line

```
rush_yrds_natty = df.iloc[164]['rush yards']
rush_yrds_natty
```