Data Series and Frames

LECTURE 6

Organization of Lecture 6

- Getting Used to Pandas Data Structures
- Reshaping Data
- Handling Missing Data
- Combining Data
- Ordering and Describing Data
- Transforming Data
- Taming Pandas File I/O

Imperfect Data

- Incorrect data, outliers, not in order, missing, etc.
- `pandas` uses `numpy.nan`
- Must take care of imperfect data
  - Can clean up files first (e.g., open .xls files)
  - Steps
    - Delete missing data (all nan values, columns with any nan values, rows with any nan values
    - Can input missing data
      - replace them with “clean” values, e.g. 0’s, 1’s, **with an average**, “undecided”, etc.

Imperfect Data

- Have panda take care of missing data with built-in functions
  - frame.dropna(how="all") or `frame.dropna(how="all", axis=1)`
  - `Frame.isnull()` or `frame.notnull()`
    - `isnull() returns True if value is a nan, False otw`
- Have panda fill in missing data with built-in functions
  - frame.fillna()
  - frame.replace(val_or_list, new_val) #if new_val is a list, must be same size
  - Both return a new frame, unless you specify `inplace = True`

Example

```
#insert a Nan first
#get series from the data frame, use loc on series
alco['Spirits'].loc['Arizona', 1979] = np.nan

#fix the dirty series by imputing the average
sp=alco['Spirits’]       #get series
clean = sp.notnull()     #clean rows in dirty column
#'-' is negation, imput clean’s mean into dirty rows
sp[-clean] = sp[clean].mean()
```

```
Arizona 1977  1.20  0.22     0.58
        1978  1.31  0.54     1.16
        1979  1.19  0.38     0.74

              Beer  Wine  Spirits
State   Year
Alabama 1977  1.20  0.22     0.58
        1978  1.31  0.54     1.16
        1979  1.19  0.38     0.74
Alaska  1977  1.25  1.22     0.32
        1978  1.31  0.54     1.16
        1979  1.17  1.38     1.74
Arizona 1977  1.20  0.22     0.58
        1978  1.31  0.54     1.16
        1979  1.19  0.38      NaN

              Beer  Wine  Spirits
State   Year
Alabama 1977  1.20  0.22     0.58
        1978  1.31  0.54     1.16
        1979  1.19  0.38     0.74
Alaska  1977  1.25  1.22     0.32
        1978  1.31  0.54     1.16
        1979  1.17  1.38     1.74
Arizona 1977  1.20  0.22     0.58
        1978  1.31  0.54     1.16
        1979  1.19  0.38     0.93
```

Combining Data

- May have data in two frames you want to combine for processing
  - e.g. compare alcohol consumption of countries by their GDPs
  - e.g. compare alcohol consumption of states by reported weather in states
  - e.g. State population with consumption of alcohol
- Pandas provides functions for merging and concatenating frames
  - Need an identical indexes (e.g. state)
  - When only one match in right frame to each row in left – one-to-one merging
  - When several in right to each row in left – one-to-many merging
  - When several matches for some rows in each frame – many-to-many
  - Any holes will be filled with numpy.nans

Merging –both frames have same column

- e.g. State population with consumption of alcohol

```
#merge both frames on the State (key) column
mf = pd.merge(regions2010, alco2010, on = 'State')
print (mf.head())
```

State	2010
West Virginia	1,854,239
Wisconsin	5,690,475
Wyoming	564,487
NaN	NaN
Puerto Rico	3,721,525

State	Beer	Wine	Spirits
Alabama	1.20	0.22	0.58
Alaska	1.31	0.54	1.16
Arizona	1.19	0.38	0.74
Arkansas	1.07	0.17	0.60
California	1.05	0.55	0.73

State	2010	Beer	Wine	Spirits
Alabama	4,785,437	1.20	0.22	0.58
Alaska	713,910	1.31	0.54	1.16
Arizona	6,407,172	1.19	0.38	0.74
Arkansas	2,921,964	1.07	0.17	0.60
California	37,319,502	1.05	0.55	0.73

Other ways to combine

- IF 2 columns have identical names, pandas adds suffixes
  - “_l” and “_r” add to column names
- If you want to merge on indexes, use optional parameters left_index=True and right_index = True
  - `mf = pd.merge(regions2010, alco2010, left_index=True, right_index=True)`
  - Result is same, but default sorting order may be different
    - order of population, not the state
- IF both indexes are designated as keys, use join
  - `regions2010.join(alco2010)`
- Then there is concat – concatenates list of frames by placing them next to each other, either vertically (axis=0, default) or horizontally (axix=1)
  - `pd.concat([regions2010, alco2010], axis=1)`

Issues with Combining

- If you are combining frames of similar content – such as populations of US and populations of Canada *(apples to apples)* – use merge or concat
- If not, use merge to combine complementary content (such as population an alcohol consumption rates *(apples to oranges)*
- Combining data may result in duplicates
  - `duplicated ([subset])` returns Boolean series if each row in all or a subset of columns is duplicated. [subset] is a list of column names
  - `drop-duplicates ([subset])` function returns copy of a frame or series with duplicates removed (may use `inplace=True`)
  - Optional parameter first or last or each True duplicate is removed.

pandas File I/O

- Can read csv, JSON, xls, text files…
- Examples:

```
#has 3 header rows to skip
regions= pd.read_csv("USPopData.csv", header=3)

#has multiple indices, no header row
alco = pd.read_csv("niaaa-report.csv", header=0, index_col=[0,1])
```

- Now, may have to “clean” up files further…