DSC 201: Data Analysis & Visualization

Aggregation

Dr. David Koop
Data Analysis Scenarios

• Often want to analyze data by some grouping:
  - Dogs vs. cats
  - Millennials vs. Gen-X vs. Baby Boomers
  - Physics vs. Chemistry

• Compute statistics based on those groupings
  - max, min, median

• Perform your own type of transformation (top-k, spread, etc.)

• Create visualizations for each group
Split-Apply-Combine

• Coined by H. Wickham, 2011
• Similar to Map (split+apply) Reduce (combine) paradigm
• The Pattern:
  1. **Split** the data by some grouping variable
  2. **Apply** some function to each group independently
  3. **Combine** the data into some output dataset
• The apply step is usually one of:
  - Aggregate
  - Transform
  - Filter
Aggregation of time series data, a special use case of groupby, is referred to as resampling in this book and will receive separate treatment in Chapter 10.

**GroupBy Mechanics**

Hadley Wickham, an author of many popular packages for the R programming language, coined the term **split-apply-combine** for talking about group operations, and I think that's a good description of the process. In the first stage of the process, data contained in a pandas object, whether a Series, DataFrame, or otherwise, is split into groups based on one or more keys that you provide. The splitting is performed on a particular axis of an object. For example, a DataFrame can be grouped on its rows (axis=0) or its columns (axis=1). Once this is done, a function is applied to each group, producing a new value. Finally, the results of all those function applications are combined into a result object. The form of the resulting object will usually depend on what's being done to the data. See Figure 9-1 for a mockup of a simple group aggregation.

Figure 9-1. Illustration of a group aggregation

Each grouping key can take many forms, and the keys do not have to be all of the same type:

- A list or array of values that is the same length as the axis being grouped
- A value indicating a column name in a DataFrame

[W. McKinney, Python for Data Analysis]
## Splitting by Variables

<table>
<thead>
<tr>
<th>name</th>
<th>age</th>
<th>sex</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>13</td>
<td>Male</td>
</tr>
<tr>
<td>Mary</td>
<td>15</td>
<td>Female</td>
</tr>
<tr>
<td>Alice</td>
<td>14</td>
<td>Female</td>
</tr>
<tr>
<td>Peter</td>
<td>13</td>
<td>Male</td>
</tr>
<tr>
<td>Roger</td>
<td>14</td>
<td>Male</td>
</tr>
<tr>
<td>Phyllis</td>
<td>13</td>
<td>Female</td>
</tr>
</tbody>
</table>

### .(sex)

<table>
<thead>
<tr>
<th>name</th>
<th>age</th>
<th>sex</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>13</td>
<td>Male</td>
</tr>
<tr>
<td>Peter</td>
<td>13</td>
<td>Male</td>
</tr>
<tr>
<td>Roger</td>
<td>14</td>
<td>Male</td>
</tr>
</tbody>
</table>

### .(age)

<table>
<thead>
<tr>
<th>name</th>
<th>age</th>
<th>sex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>15</td>
<td>Female</td>
</tr>
<tr>
<td>Alice</td>
<td>14</td>
<td>Female</td>
</tr>
<tr>
<td>Roger</td>
<td>14</td>
<td>Male</td>
</tr>
</tbody>
</table>

*Figure 4: Two examples of splitting up a data frame by variables. If the data frame was split up by both sex and age, there would only be one subset with more than one row: 13-year-old males.*

### Table 3: Summary of processing function restrictions and null output values for all output types. Explained in more detail in each output section.

3.2. Output

The output type defines how the pieces will be joined back together and how they will be labelled. The labels are particularly important as they allow matching up of input and output.

The input and output types are the same, except there is an additional output data type, _, which discards the output. This is useful for functions like `plot()` and `write.table()` that are called only for their side effects, not their return values.

The output type also places some restrictions on what type of results the processing function should return. Generally, the processing function should return the same type of data as the eventual output, (i.e., vectors, matrices and arrays for *aply* and data frames for *dply*), but some other formats are accepted for convenience and are described in Table 3. These are explained in more detail in the individual output type sections.

### Output: Array (*aply*)

With array output the shape of the output array is determined by the input splits and the dimensionality of each individual result. Figures 5 and 6 illustrate this pictorially for simple splitting by variables.
## Apply+Combine: Counting

### Table Examples

<table>
<thead>
<tr>
<th>sex</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>3</td>
</tr>
<tr>
<td>Female</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>age</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>3</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sex</th>
<th>age</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>Male</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>Female</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>Female</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>Female</td>
<td>15</td>
<td>1</td>
</tr>
</tbody>
</table>

[H. Wickham, 2011]
In Pandas

- `groupby` method creates a `GroupBy` object
- `groupby` doesn't actually compute anything until there is an aggregation or we wish to examine the groups
- Choose keys (columns) to group by
Example: Tipping Data

- http://www.cis.umassd.edu/~dkoop/dsc201-2016fa/notebooks/tipping.ipynb
Tableau Contest

- https://public.tableau.com/s/Student-Viz-Assignment-Contest
Other Operations

• Quantiles: return values at particular splits
  - Median is a 2-quantile
  - `df.quantile(0.1)`
  - also works on groups

• Can return data from group-by without having the keys in the index
  `(as_index=False)` or use `reset_index` after computing

• Grouped weighted average via apply
## Pivot Tables

- Data summarization tool in many spreadsheet programs
- Aggregates a table of data by one or more keys with some keys arranged on rows (index), others as columns (columns)
- Pandas supports via `pivot_table` method
- `margins=True` gives partial totals
- Can use different aggregation functions via `aggfunc` kwarg

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>values</td>
<td>Column name or names to aggregate. By default aggregates all numeric columns</td>
</tr>
<tr>
<td>rows</td>
<td>Column names or other group keys to group on the rows of the resulting pivot table</td>
</tr>
<tr>
<td>cols</td>
<td>Column names or other group keys to group on the columns of the resulting pivot table</td>
</tr>
<tr>
<td>aggfunc</td>
<td>Aggregation function or list of functions; 'mean' by default. Can be any function valid in a groupby context</td>
</tr>
<tr>
<td>fill_value</td>
<td>Replace missing values in result table</td>
</tr>
<tr>
<td>margins</td>
<td>Add row/column subtotals and grand total, False by default</td>
</tr>
</tbody>
</table>

See Table 9-2 for a summary of `pivot_table` methods.

### Cross-Tabulations: Crosstab

A cross-tabulation (or `crosstab` for short) is a special case of a pivot table that computes group frequencies. Here is a canonical example taken from the Wikipedia page on cross-tabulation:

```python
In [292]: data
Out[292]:
Sample  Gender    Handedness
0       1  Female  Right-handed
1       2    Male   Left-handed
2       3  Female  Right-handed
3       4    Male  Right-handed
4       5    Male   Left-handed
5       6    Male  Right-handed
6       7  Female  Right-handed
7       8  Female   Left-handed
8       9    Male  Right-handed
9      10  Female  Right-handed

As part of some survey analysis, we might want to summarize this data by gender and handedness. You could use `pivot_table` to do this, but the `pandas.crosstab` function is very convenient:

```python
In [293]: pd.crosstab(data.Gender, data.Handedness, margins=True)
Out[293]:
         Handedness
Gender       Left-handed  Right-handed  All
Female            1             4    5
Male              2             3    5
All                3             7   10
```

The first two arguments to `crosstab` can each either be an array or Series or a list of arrays. As in the tips data:

```python
In [294]: pd.crosstab([tips.time, tips.day], tips.smoker, margins=True)
```

[W. McKinney, Python for Data Analysis]
Crosstabs

- `crosstab` is a special case for group frequencies (aggfunc='count')

  ```
  In [293]: pd.crosstab(data.Gender, data.Handedness, margins=True)
  Out[293]:
   Handedness    Left-handed  Right-handed  All
  Gender          Female          1            4     5
                  Male            2            3     5
                All            3            7    10
  ```

- Tipping example
- Also see the Federal Election Database example in the book
How Fake News Goes Viral

Anti-Trump protestors in Austin today are not as organic as they seem. Here are the busses they came in. #fakeprotests
#trump2016 #austin
More on Fake News

• Just how partisan is Facebook's fake news? We tested it
• Paul Horner creates fake news websites
  - "Honestly, people are definitely dumber. They just keep passing stuff around. Nobody fact-checks anything anymore…"
• Google has had its own issues:

In the news

FINAL ELECTION 2016 NUMBERS: TRUMP WON BOTH POPULAR (62.9 M -62.2 M ) AND ELECTORAL COLLEGE ...
70news - WordPress.com - 2 days ago
thump headshot ...

Clinton vs. Trump Popular Vote: Are There Still Uncounted Ballots?
Heavy.com - 23 hours ago
Laramie County 2016 Election Results Final
Kgab - 23 hours ago
Translation Bias

- Try the following Turkish to English translation (o is a gender-neutral pronoun) on Google Translate
  - o bir doktor
  - o bir hemşire
  - [via #FATML tweet]

- Similar results for Finnish to English…

- Paper by Caliskan-Islam et al. on algorithms learning language from corpora also learning biases
Occupation-Gender Association

Figure 1. Occupation-gender association
Pearson’s correlation coefficient $\rho = 0.90$ with $p$-value $< 10^{-18}$.

[Caliskan-Islam et al., 2016]
Who or what is at fault?
What should data scientists be doing?