Data Wrangling with pandas Cheat Sheet

http://pandas.pydata.org

Tidy Data – A foundation for wrangling in pandas

Tidy data complements pandas’s vectorized operations. Pandas will automatically preserve observations as you manipulate variables. No other format works as intuitively with pandas.

Syntax – Creating DataFrames

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>9</td>
</tr>
</tbody>
</table>

`df = pd.DataFrame(
    {"a": [4, 5, 6],
     "b": [7, 8, 9],
     "c": [10, 11, 12]},
    index = [1, 2, 3])`

Specify values for each row.

Reshaping Data – Change the layout of a data set

<table>
<thead>
<tr>
<th>df.melt(df)</th>
<th>df.pivot(columns='var', values='val')</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gather columns into rows.</td>
<td>Spread rows into columns.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>df.concat([df1, df2])</th>
<th>df.concat([df1, df2], axis=1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Append rows of DataFrames</td>
<td>Append columns of DataFrames</td>
</tr>
</tbody>
</table>

Subset Observations (Rows)

<table>
<thead>
<tr>
<th>df[df.Length &gt; 7]</th>
<th>df.sample(frac=0.5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract rows that meet logical criteria.</td>
<td>Randomly select fraction of rows.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>df.drop_duplicates()</th>
<th>df.nsmallest(n, 'value')</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove duplicate rows (only considers columns).</td>
<td>Select and order bottom n entries.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>df.head(n)</th>
<th>df.tail(n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select first n rows.</td>
<td>Select last n rows.</td>
</tr>
</tbody>
</table>

Subset Variables (Columns)

<table>
<thead>
<tr>
<th>df[[ 'width', 'length', 'species']</th>
<th>df.filter(regex='regex')</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select multiple columns with specific names.</td>
<td>Select columns whose name matches regular expression regex.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>df['width'] or df.width</th>
<th>df.filter(regex='regex')</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select single column with specific name.</td>
<td>Select columns whose name matches regular expression regex.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>df.loc[:,'x2':'x4']</th>
<th>df.loc[df[ 'a' ] &gt; 10, ['a', 'c']]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select all columns between x2 and x4 (inclusive).</td>
<td>Select rows meeting logical condition, and only the specific columns.</td>
</tr>
</tbody>
</table>

Method Chaining

Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

`df = (pd.melt(df).rename(columns={
    'variable' : 'var',
    'value' : 'val'})
    .query('val > 200'))

Logic in Python (and pandas)

- `<` Less than
- `>` Greater than
- `<=` Less than or equal to
- `>=` Greater than or equal to
- `==` Equal to
- `!=` Not equal to
- `==` Group membership
- `is` Is NaN
- `isnot` is not NaN
- `&` Logical and
- `|` Logical or
- `^` Logical xor

regex (Regular Expressions) Examples

- `\.` Matches strings containing a period `.`
- `\d*` Matches strings ending with word 'Length'
- `\d+` Matches strings beginning with the word 'Sepal'
- `\d+\d+\d+\d+\d+` Matches strings beginning with 'x' and ending with 1,2,3,4,5
- `\d+\d+\d+\d+\d+` Matches string that excludes the string 'Species'

**Summarize Data**

- `df['w'].value_counts()` - Count number of rows with each unique value of variable `w`
- `len(df)` - # of rows in DataFrame.
- `df['w'].unique()` - # of distinct values in a column.
- `df.describe()` - Basic descriptive statistics for each column (or GroupBy)

Pandas provides a large set of summary functions that operate on different kinds of pandas objects (DataFrame columns, Series, GroupBy, Expanding and Rolling (see below)) and produce single values for each of the pandas groups. When applied to a DataFrame, the result is returned as a pandas Series for each group. Examples:

- `sum()` - Sum values of each object.
- `count()` - Count non-NA/null values of each object.
- `median()` - Median value of each object.
- `quantile([0.25, 0.75])` - Quantiles of each object.
- `apply()` - Apply function to each object.
- `min()` - Minimum value in each object.
- `max()` - Maximum value in each object.
- `mean()` - Mean value of each object.
- `var()` - Variance of each object.
- `std()` - Standard deviation of each object.

**Handling Missing Data**

- `df.dropna()` - Drop rows with any column having NA/null data.
- `df.fillna(value)` - Replace all NA/null data with value.

**Group Data**

- `df.groupby(by="col")` - Return a GroupBy object, grouped by values in column named "col".
- `df.groupby(level="ind")` - Return a GroupBy object, grouped by values in index level named "ind".

All of the summary functions listed above can be applied to a group. Additional GroupBy functions:

- `size()` - Size of each group.
- `agg(function)` - Aggregate group using function.

The examples below can also be applied to groups. In this case, the function is applied on a per-group basis, and the returned vectors are of the length of the original DataFrame.

- `shift(1)` - Copy with values shifted by 1.
- `rank(method='dense')` - Ranks with no gaps.
- `rank(method='min')` - Ranks. Ties get min rank.
- `rank(pct=True)` - Ranks rescaled to interval [0, 1].
- `rank(method='first')` - Ranks. Ties go to first value.

**Make New Columns**

- `df.assign/Area=lambda df: df.Length*df.Height` - Compute and append one or more new columns.
- `pd.qcut(df.col, n, labels=False)` - Bin column into n buckets.

**Windows**

- `df.expanding()` - Return an Expanding object allowing summary functions to be applied cumulatively.
- `df.rolling(n)` - Return a Rolling object allowing summary functions to be applied to windows of length n.

**Combine Data Sets**

- `pd.merge(adf, bdf, how='left', on='x1')` - Join matching rows from bdf to adf.
- `pd.merge(adf, bdf, how='right', on='x1')` - Join matching rows from adf to bdf.
- `pd.merge(adf, bdf, how='inner', on='x1')` - Join data. Retain only rows in both sets.
- `pd.merge(adf, bdf, how='outer', on='x1')` - Join data. Retain all values, all rows.

**Plotting**

- `df.plot.hist()` - Histogram for each column.
- `df.plot.scatter(x='w', y='h')` - Scatter chart using pairs of points.

**Handling Missing Data**

- `df.dropna()` - Drop rows with any column having NA/null data.
- `df.fillna(value)` - Replace all NA/null data with value.

**Combine Data Sets**

- `pd.merge(adf, bdf, on='x1')` - Join matching rows from bdf to adf.
- `pd.merge(adf, bdf, how='outer', on='x1')` - Join data. Retain all values, all rows.

**Plotting**

- `df.plot.hist()` - Histogram for each column.
- `df.plot.scatter(x='w', y='h')` - Scatter chart using pairs of points.