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.

Syntx – Creating DataFrames

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

Specify values for each column.

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

Specify values for each row.

Reshaping Data – Change the layout of a data set

```
df.sort_values('mpg')  # Order rows by values of a column (low to high).
df.sort_values('mpg', ascending=False)  # Order rows by values of a column (high to low).
df.rename(columns = {'y':'year'})  # Rename the columns of a DataFrame
```

Subset Observations (Rows)

```
df[df.Length > 7]  # Extract rows that meet logical criteria.
```

```
df.drop_duplicates()  # Remove duplicate rows (only considers columns).
```

```
df.head(n)  # Select first n rows.
df.tail(n)  # Select last n rows.
```

```
df.sample(frac=0.5)  # Randomly select fraction of rows.
df.sample(n=10)  # Randomly select n rows.
df.iloc[10:20]  # Select rows by position.
```

```
df.nlargest(n, 'value')  # Select and order top n entries.
df.nsmallest(n, 'value')  # Select and order bottom n entries.
```

Subset Variables (Columns)

```
df[[ 'width', 'length', 'species']])  # Select multiple columns with specific names.
df[ 'width'] or df.width  # Select single column with specific name.
```

```
df.filter(regex = 'regex')  # Select columns whose name matches regular expression regex.
```

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)  # Gather columns into rows.
       .rename(columns={
           'variable' : 'var',
           'value' : 'val'})  # Save in its own column
       .query('val > 200'))  # Only select values.
```

Logic in Python (and pandas)

```
< Less than  
> Greater than  
== Equals  
<= Less than or equals  
>= Greater than or equals  
& Logical and  
| Logical or  
^ Logical xor
```

```
Is NaN  
Is not NaN
```

```
df['variable'].isin(values)  # Group membership
```

```
pd.isnull(obj)  # Not equal to
```

```
pd.notnull(obj)  # Logical and, or, not, xor, any, all
```

```
df.column.isin()  # Logical and, or, not, xor, any, all
```

Rstudio - Cheatsheet
This cheatsheet inspired by Rstudio Data Wrangling Cheatsheet. Written by Irv Lustig, Princeton Consultants.
### 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 groups. When applied to a DataFrame, the result is returned as a pandas Series for each column. 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.
- `mean()`: Mean value of each object.
- `min()`: Minimum value in each object.
- `max()`: Maximum value in 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.

**pandas** provides a large set of **vector functions** that operate on all columns of a DataFrame or a single selected column (a pandas Series). These functions produce vectors of values for each of the columns, or a single Series for the individual Series. Examples:

- `max(axis=1)`: Max value in each object.
- `min(axis=1)`: Min value in each object.
- `var()`: Variance of each object.
- `std()`: Standard deviation of each object.

### 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.

### 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.

### 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.

### Plotting

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

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.
- `cumsum()`: Cumulative sum.
- `cummax()`: Cumulative max.
- `cummin()`: Cumulative min.
- `cumprod()`: Cumulative product.

### Combine Data Sets

<table>
<thead>
<tr>
<th>df</th>
<th>bdf</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
</tr>
</tbody>
</table>

**Join data. Retain only rows in both sets.**

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

**All rows in adf that have a match in bdf.**

- `adf[adf.x1.isin(bdf.x1)]`: All rows in adf that have a match in bdf.
- `~adf.x1.isin(bdf.x1)]`: All rows in adf that do not have a match in bdf.

**Rows that appear in both ydf and zdf (Intersection).**

- `pd.merge(ydf, zdf)`: Rows that appear in both ydf and zdf (Intersection).

**Rows that appear in either or both ydf and zdf (Union).**

- `pd.merge(ydf, zdf, how='outer')`: Rows that appear in either or both ydf and zdf (Union).

http://pandas.pydata.org/ This cheat sheet inspired by Rstudio Data Wrangling Cheatsheet. Written by Irv Lustig, Princeton Consultants.