# Python vs R

*Cho-Yi Chen*

NTU.Joey@Gmail.com

*Beta* 0.1

# Language

| Action | Python | R |
|---|---|---|
| Comment | # | # |
| Logic | True/False | TRUE/FALSE; T/F |
| String | 'x', "x", """x""" | 'x', "x" |
| Help | help(obj), ?obj[1], obj?[1] | help(obj), ?obj |
| Example | | example(obj), demo(obj) |
| Import package | import pkg | library("pkg name") |
| If…Else | if x < 0:<br>  print('Negative')<br>elif x == 0:<br>  print('Zero')<br>else:<br>  print('Positive') | if(x < 0){<br>  print('Negative')<br>} |
| For loop | for i in range(1, 6):<br>  print(i) | for(i in 1:5){<br>  print(i)<br>} |
| Map/Apply | map(function, iterable, ...) | apply(X, MARGIN, FUN, ...)<br>by(data, INDICES, FUN, ...) |

# Input and Output

| Action | Python | R |
|---|---|---|
| Prompt user input | input(), raw_input() | readline(prompt = "") |

# String

| Action | Python | R |
|---|---|---|
| Concatenate | str.join()[2] | paste(..., sep = " ") |

[1] iPython only

| | |
|---|---|
| Concatenate and Print | `cat(..., file="", sep=" ", append=FALSE)` |

# Data Type

| Action | Python | R |
|---|---|---|
| Type/Class | `type(x)` | `class(x)` |
| Type checking | `isinstance(x)` | |
| Type transform | `y = int(x)` | `Y = as.integer(x)` |
| Delete | `del x` | `rm(x)` |

# Container[3]

| Action | Python | R |
|---|---|---|
| **Array[4]/Vector** | | |
| Create | `x = np.array([1,2,3])` | `x = c(1,2,3)` |
| From range | `y = np.arange(4, 7)` | `y = 4:6` |
| Concatenation | `np.concatenate((x,y))` | `c(x,y)` |
| Math operation | `x + y` | `x + y` |
| **Matrix** | `np.matrix()` | `matrix()` |
| **Series/List[5]** | | |
| Create | `s = pd.Series({'L1': x, 'L2': y})` | `s = list(L1=x, L2=y, L3=z)` |
| Get the 1st element[6] | `s[0], s['L1']` | `s[1], s$L1` |
| **DataFrame/data.frame** | | |
| Create | `df = pd.DataFrame(data)` | `df = data.frame(data)` |
| Get the 1st column | `df.c1, df['c1']` | `df$c1, df[,"c1"], df[,1]` |
| Get the 1st row | `df.iloc[0], df.loc['r1']` | `use slice` |
| Slice the first 2 columns | `df.iloc[:,:2], df.loc[:,['c1', 'c2']]` | `df[1:2], df[c('c1', 'c2')]` |
| Slice the first 2 rows | `df.iloc[:2], df.loc[['r1','r2']` | `df[1:2,]` |
| Slice a block | `df.iloc[:2, :2]` | `df[1:2, 1:2]` |
| Head/tail | `df.head()/df.tail()` | `head(df)/tail(df)` |
| Dataframe I/O | I/O tools (Support Excel) | |
| From text file (or URL)[7] | `pd.read_table()` | `read.table()` |

---

[2] E.g., `' '.join(['Hello', 'World!'])`

[3] Setup in Python: `import numpy as np; import pandas as pd`

[4] Numpy ndarray can be multi-dimensional, structured, and nested

[5] Elements can be any kinds of objects

[6] Python index starts from `0`, whereas R index starts from `1`

[7] Input can be a file path, URL, or file-like object. For csv file, `read_csv()` and `read.csv()` are also available.

| | | |
|---|---|---|
| From clipboard | `pd.read_clipboard()` | `readClipboard()` |
| From html | `pd.read_thml()` | |
| To text file | `df.to_csv()` | `write.table()` |
| To html | `df.to_html()` | |
| Concatenate | `pd.concat()` | |
| Join | `pd.merge()` | |
| Pivot table | `pd.pivot_table()` | |
| Info/Structure/Data type | `df.info(), df.dtypes` | `str(df)` |
| Remove NaN | `df.dropna()` | |
| Sort | `df.sort_index()` | |

# Statistics[8]

| Action | Python | R |
|---|---|---|
| **Descriptive statistics** | | |
| Summary | `sp.stats.describe()` | `summary()` |
| Mean | `np.mean()` | |
| Median | `np.median()` | |
| Variance | `np.var()` | `var()` |
| Standard deviation | `np.std()` | |
| **Correlation** | | |
| Pearson correlation coefficient | `sp.stats.pearsonr()` | `cor.test()` |
| Spearman's rho | `sp.stats.spearmanr()` | `cor.test()` |
| Kendall's tau | `sp.stats.kendalltau()` | `cor.test()` |
| Correlation matrix | `np.corrcoef(), df.corr()` | `cor()` |
| Covariance matrix | `np.cov()` | `cov()` |
| Partial Correlation | ? | package 'ppcor' |
| **Multiple Corrections** | | |
| FDR (BH) | `statsmodels.multipletests()` | `p.adjust()` |
| Q-value | | `qvalue()`[9] |
| **Contingency table** | | |
| Chi-square test | `sp.stats.chi2_contingency()`[10] | |
| Fisher's exact test | `sp.stats.fisher_exact()` | |
| **Parametric statistics** | | |
| Student's *t*-test | | |
| One-sample *t*-test | `sp.stats.ttest_1samp()` | |
| Independent two-sample t-test | `sp.stats.ttest_ind()` | |

[8] Setup in Python: `import scipy as sp`
[9] From Bioconductor or from storeylab
[10] See also `contingency.expected_freq`, `fisher_exact`, `chisquare`

| | |
|---|---|
| Dependent *t*-test (paired samples) | `sp.stats.ttest_rel()` |
| **Nonparametric Statistics** | |
| Wilcoxon Rank Sum Test | `sp.stats.ranksums()`[11] |
| Wilcoxon Signed-Rank Test | `sp.stats.wilcoxon()` |
| Mann-Whiney U Test | `sp.stats.mannwhitneyu()`[12] |
| Kolmogorov-Smirnov test | |
| For goodness of fit | `sp.stats.kstest()` |
| For 2 independent samples | `sp.stats.ks_2samp()` |

# Plotting[13]

| Action | Python | R |
|---|---|---|
| **Figure Types** | | |
| Heatmap | `plt.pcolor`[14]`()` | `heatmap()` |
| Line plot | `plt.plot(), df.plot()` | `plot()` |
| Bar plot | `df.plot(kind="bar")` | |
| Box plot | `plt.boxplot(), df.boxplot()` | |
| Histogram[15] | `plt.hist(), df.hist()` | `hist()` |
| Table | `plt.table()` | |
| Animation | See Howto | |
| **Supplementary Elements** | | |
| Kernel density estimation[16] | `scipy.stats.gaussian_kde()`[17] | `density()` |
| Area fill | `plt.fill()` | `polygon()` |
| Rug plot (Carpet plot) | `plt.plot(x, [0]*len(x), '+')` | |
| Error bar | `plt.errorbar()` | |
| Horizontal line | `plt.axhline()` | |
| Vertical line | `plt.axvline()` | |
| **Text & Anotations** | | |
| Text | `plt.text()` | |
| Mathematical expression | e.g., `r'$\sigma_i=15$'` | |
| Annotation text[18] | `plt.annotate()` | |
| Annotation axes | See annotation guide | |

[11] This test should be used to compare two samples from continuous distributions. It does not handle ties between measurements in `x` and `y`. For tie-handling and an optional continuity correction, see `stats.mannwhitneyu`.

[12] The reported p-value is for a one-sided hypothesis, to get the two-sided p-value multiply the returned p-value by 2.

[13] Setup in Python: `import matplotlib.pyplot as plt`.

[14] For huge map, use `pcolormesh` or `imshow` instead. Note that R's `heatmap` implements hierarchical clustering.

[15] Histograms can be a poor method for determining the shape of a distribution because it is so strongly affected by the number of bins used.

[16] Kernel density plots are usually a much more effective way to view the distribution of a variable.

[17] See also http://docs.scipy.org/doc/scipy/reference/tutorial/stats.html#kernel-density-estimation

[18] Text w/ arrow

# Plotting auxiliary

| Action | Python | R |
|---|---|---|
| **Figure** | | |
| Create a figure instance | `plt.figure()` | |
| Set the size | `plt.figure(figsize=(x,y))` | |
| Get current figure instance | `plt.gcf()` | |
| Clear the current figure | `plt.clf()` | |
| Close a figure | `plt.close()` | |
| Figure subplots[19] | `plt.subplot()` | |
| Figure layout/spacing | `fig.subplots_adjust()` | |
| | `fig.add_axes([L,B,W,H])` | |
| | `plt.tight_layout()`[20] | |
| **Axes**[21] | | |
| Create an axes | `plt.axes()` | |
| Get current axes | `plt.gca()` | |
| Clear the current axes | `plt.cla()` | |
| Axes title | `plt.title()` | |
| Axes xlabel | `plt.xlabel()` | |
| Axes ylabel | `plt.ylabel()` | |
| Axes legend | `plt.legend()` | |
| Axes grid | `plt.grid()` | |
| Axes ticks | `plt.xticks(), plt.yticks()` | |
| Axes ratio | `ax.set_aspect()`[22] | |
| **Axis** | | |
| Get current axis instance[23] | `ax.xaxis, ax.yaxis` | |
| Control the tick side | `ax.yaxis.tick_left()`[24] | |
| Control the tick direction | `ax.xaxis.set_tick_params()`[25] | |
| Hide axis | `ax.xaxis.set_visible(False)` | |
| Multiple y-axis | `twinx(), twiny(), see Howto` | |
| **Tick** | | |
| Set tick linewidths | See howto | |
| Set tick direction | `ax.xaxis.set_tick_params()` | |
| Set tick labels | `ax.set_xticklabels()` | |
| **Object** | Return from plotting function | |

[19] Rectangular grid axes
[20] Automatically adjusts subplot params so that the subplot(s) fits in to the figure area. See Tight Layout Guide
[21] Default axes will be created w/ `figure()` or `subplot()`. User can specify the location of axes by using `axes()`.
[22] Can also be set when calling `add_subplot()`, e.g., `fig.add_subplot(111, aspect='equal')`
[23] `ax` is an axes instance
[24] See also `ax.xaxis.tick_bottom()`, …, etc
[25] Using the parameter: `direction`, could be '`in`' or '`out`'.

| | |
|---|---|
| Object property getter | 1. `obj.get_<tab>`<br>2. `plt.getp()` |
| Object property setter | 1. `obj.set_<tab>`<br>2. `plt.setp()` |
| **Configuration** | |
| Global configuration[26] | `matplotlib.rcParams`<br>`matplotlib.rc()` |

[26] Matplotlib uses matplotlibrc configuration files to customize all kinds of properties.