Python Course

Kevin Sheppard

Monday 10th February, 2014

Contents

1 Getting Started

Task 1.1. Install Anaconda on your computer.

- Download the Anaconda installer from Continuum Analytics.
- Install the environment using the downloaded file.

Task 1.2. Install a good text editor if you don't have one

- Sublime Text 3
  - Perpetual nagware, not free ($70)
  - Python at its core, many Python-specific add-ons
- Free alternatives
  - Windows: Notepad++

  - Linux: gedit or similar
  - OSX: TextWrangler or TextMate+Python Bundle

Task 1.3. Setup a virtual environment.

- Use `conda` to setup the virtual environment *econ*
- Add packages to the virtual environment using `conda`
- Update the virtual environment using `conda`
- Activate the virtual environment
  - `activate econ` on Windows
  - `source activate econ` on nix
- Install additional packages using `pip`

Task 1.4. Open IPython in the terminal.

- Open a terminal (nix) or cmd (Windows) and activate the environment
- Start IPython using `ipython`
- Entering commands
- Configuration options
  - pylab
  - Customizing using configuration files

Task 1.5. Open IPython QtConsole.

- Open a terminal (nix) or cmd (Windows) and activate the environment
- Start IPython using `ipython-qtconsole`
- Configuration options
  - `pylab=inline`
  - Customizing using configuration files
- Cell mode
- History
- Magics and special functions
  - Examples: `ls`, `lsmagic`, `timeit`
- Help

Task 1.6. Open IPython Notebook.

- Open a terminal (nix) or cmd (Windows) and activate the environment
- If you don't have Chrome or Firefox installed, install one of these

- Installing MathJax locally
- Start IPython using `ipython-notebook`
- Configuration options
  - Command line switches
  - Magic keywords: `pylab`, `pylab=inline`, `matplotlib=inline`
- Alternative modes
  - Headings
  - Markdown
    * Emphasis
    * Code
    * Math
  - Code

Task 1.7. Create launchers and links for the virtual environment, IPython, IPython Qtconsole, IPython Notebook and Spyder.

- Maybe Windows only

Task 1.8. Open Spyder

- Open a terminal (nix) or cmd (Windows) and activate the environment
- Start Spyder using `spyder`
- Overview of windows

Task 1.9. Test the environment

- Run some commands to test the environment
  - NumPy
  - SciPy
  - Matplotlib
  - Pandas
  - Statsmodels

2 Basic Data Types

Immutable Types

Task 2.1. Integers

- Entering
- Variable names
  - Legal names
  - Common naming conventions
- Assignment
  - Multiple assignment
  - Split assignment

Task 2.2. Basic floats

- Entering
- Division warning
  - future imports

Task 2.3. Basic complex numbers

- Entering

Task 2.4. Basic Long Integers

- Entering
  - `type`
- Math

Task 2.5. Basic math and mixing types

Task 2.6. Basic Boolean data and None

Task 2.7. Basic strings

- Entering: 'text' or "text"
- Adding
- Multiplying
- str() on other types

Task 2.8. Tuples

- Entering
  - (,) or tuple

Task 2.9. The xrange type

- The old (*and new*) version range

Task 2.10. Variable Entry

- Assign the following values to variable, and predict the type of the variable

2	Python	True
3.14	is fun	None
3.0	is hard	(1, 2, 3)
$2+3i$	316, 488, 210, 664	

- Which of the above can be added to each other?
- For those which can be added, predict the type of the result. Verify this using type
- Use the native date type functions, e.g. int() , float() , str() , complex(), long(), bool(), tuple() etc. to convert those which can be converted.

Mutable Types

Task 2.11. Understanding Mutability

- What's the difference?

Task 2.12. Using list (perhaps the most important native Python type)

- Creating
  - [] or list
- Copying
- Accessing using slicing
  - Works similarly for strings
- Appending
- Removing
- Nested lists
  - Copying using deepcopy
  - Slicing
  - Examples and risks of mutability

Task 2.13. Input the following 1-dimesnaional list

3.14	2.71	1.61	0	1	1.57	1.41	1.73

Task 2.14. Use slicing to extract

1. The first half of the list
2. Every second element
3. The last element
4. All but the last

5. The reversed list
6. A copy of the list
   Note: see also len() and floor()
7. Delete 0 from the list
8. Append 0 at the end of the list
9. Insert 0 at the start of the list

Task 2.15. Input the following 2-dimesnaional list where the rows are the *outer* list

0	1	0
45	.7071	.7071
90	0.5	.8660
135	-.7071	.7071

Task 2.16. Use slicing to extract

1. The last row
2. The final two rows
3. The 2nd and 4th row
4. The 4,3 element (Hint, there are two ways)

Task 2.17. Input the variable
ex = 'Python is an interesting and useful language for numerical computing!'
Using slicing, extract:

1. Python
2. !
3. computing
4. `in`
   Note: There are multiple answers for all.
5. !gnitupmoc laciremun rof egaugnal lufesu dna gnitseretni na si nohtyP' (Reversed)
6. nohtyP
7. Pto sa neetn n sfllnug o ueia optn!

Task 2.18. Using Dictionaries

- Creating
  - {} or dict
- Accessing
- Adding
- Deleting

Task 2.19. Input the following dictionary where the first column contains keys, and the second contains values using three different methods.

Country	'US'
GDP	15.5
GDFDEF	237.5

1. Extract the value of GDP
2. Delete GDP from the dictionary

3 Core Numeric Data Types

Task 3.1. Arrays: array

- 1-dimensional arrays
- 2-dimensional arrays
- Differences between 1-dimensional and 2-dimensional arrays
- Higher dimensional arrays
- Array data types
- Understanding how array data is stored in the computer

Task 3.2. Create (float) arrays for the following shapes

$$x = [3\ 2\ 1], \quad y = \begin{bmatrix} 9 \\ 8 \\ 7 \end{bmatrix}, \quad z = \begin{bmatrix} 3.1 & 3.4 & 3.7 \\ 3.2 & 3.5 & 3.8 \\ 3.3 & 3.6 & 3.9 \end{bmatrix}$$

Food for thought: How many dimensions does x have? What about y?

Task 3.3. Create an 3-dimensional array with 2 panels, where each panel is z

Task 3.4. How could the previous arrays be forced to use `int` or `complex` types?

Task 3.5. Matrices: matrix

- Construction
- Difference from arrays

Task 3.6. Enter x, y and z as matrices. How are these different from the arrays entered previously? (Use `x_m` in place of x so that both are still in memory)

Task 3.7. Views can be used to *efficiently* convert between matrices and arrays

- Array to matrix, asmatrix or mat
- Matrix to array, asarray
  - Highly useful on other types such as pandas DataFrame

4 Working With Arrays

Task 4.1. Array Slicing

- Using the arrays constructed in previous tasks (x, y and z)
- Pull out the last elements of x and y
- Reverse the elements
- Pull out the lower 2×2 in z
- Pull out the 2nd row of z
- Pull out the 2nd column of z

Task 4.2. Flat Slicing

- Extract the elements below using 2-dimensional indexing and `flat` (Note: Linear algebra indices)
  - $(1, 1)$
  - $(1, 2)$
  - $(2, 1)$
  - $(3, 3)$

Task 4.3. Array math

- Addition, subtraction
- Element-by-element multiplication and division
- Dot product (matrix multiplication)
- Exponentiation
  - Exponentiation of square arrays (linear algebra)
- Modulo (remainder)
- Transpose

Task 4.4. Matrix math

- Differences from 2-dimensional arrays

Task 4.5. Array math

- Using x, y and z, try main math operations $+$, $-$, $\times$, $\div$, and `mod` using arrays with the same size
- What happens when using $\times$ on matrices?
- What happens when mixing a matrix with an array in a math operation?

Task 4.6. Broadcasting

- The rules of broadcasting
  - Examples

- Why use broadcasting?

Task 4.7. Broadcastability

Suppose the following arrays have the indicated shape. Which are broadcastable?

		a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*
		`()`	`(6,)`	`(1,6)`	`(6,1)`	`(1,3,6)`	`(6,3)`	`(3,)`	`(1,)`	`(1,1)`
a	`()`									
b	`(6,)`									
c	`(1,6)`									
d	`(6,1)`									
e	`(1,3,6)`									
f	`(6,3)`									
g	`(3,)`									
h	`(1,)`									
i	`(1,1)`									

Task 4.8. Broadcasting examples

- Can x and y be used in a mathematical expression?
- What about x and z or y and z?
- What happens when you add and subtract a larger array to a smaller array that is broadcastable?

Task 4.9. Broadcast Assignment

- What shape is $z[0]$? What about $z[:, 0]$? What about $z[:, :1]$?
- Can x or y be assigned to $z[0]$?
- Can x or y be assigned to $z[:, :1]$?
- When can a scalar be assigned to a slice?

Task 4.10. Array manipulation

- Concatenating
  - concatenate
    - hstack and vstack
    - tile
  - Slicing
    - Simple slicing
    - Numeric indexing
  - Reshaping
    - shape and reshape
    - None
    - squeeze
    - flat , flatten
  - Information about size
    - size

    - ndim
  - Memory management: Copies vs. views
    - copy
    - +0.0
    - array or matrix

Task 4.11. Array Manipulation

- Using the arrays constructed in previous tasks (x, y and z)
  - Determine the number of dimensions of each array
  - Use the previous to get the size of the last dimension
  - Create a 3-dimensional array with 3 panels where each panel is z
    - tile
    - reshape and flat (more typing)
  - Assign y to ya and verify that these are the same
  - Copy y to yc and change elements to verify that it is different
  - How can prod be used with shape to produce size?
  - How can len be used with shape to produce ndim?
  - Change the shape of x from 1-dimensional to 2 using (Assign each to a different variable):
    - reshape
    - [:, None]
    - shape
    - Which of these copies the data?

Task 4.12. Operator precedence

- Build an example to verify precedence of parentheses, exponentiation, multiplication, unary plus and subtraction (scalars are fine)

Operator	Name	Rank
( ), [ ], (,)	Parentheses, Lists, Tuples	1
**	Exponentiation	2
~	Bitwise NOT	3
+,−	Unary Plus, Unary Minus	3
*, /, //, %	Multiply, Divide, Modulo	4
+,−	Addition and Subtraction	5
&	Bitwise AND	6
^	Bitwise XOR	7
|	Bitwise OR	8
<, <=, >, >=	Comparison operators	9
==, !=	Equality operators	9
in, not in	Identity Operators	9
is, is not	Membership Operators	9
not	Boolean NOT	10
and	Boolean AND	11
or	Boolean OR	12
=,+=,−=,/=,*=,**=	Assignment Operators	13

5 Flow Control

Task 5.1. Using standard_normal to generate standard normals, simulate

$$y_t = 1.2y_{t-1} - 0.2y_{t-2} + \epsilon_t$$

Task 5.2. Compute the dot (matrix) produce of 2 arrays with arbitrary dimension using nested for loops.

- Compare the time required to using dot using %timeit

6 Combined Flow Control and List Comprehensions

Task 6.1. Using the search query data from my website, write a nested for loop that will find all words with 5 or more characters.

Task 6.2. Repeat the previous problem using only xrange as the iterator.

Task 6.3. A standard normal cdf can be computed using scipy.stats.norm.cdf. Write a while loop that can invert the normal cdf to a tolerance of 10^{-8} for any value in $(0,1)$.

Task 6.4. Simulate a SETAR which is defined as

$$\begin{aligned} y_t &= y_{t-1} + \epsilon_t \text{ if } |y_{t-1}| < 5 \\ &= 0.9 y_{t-1} + \epsilon_t \text{ otherwise} \end{aligned}$$

where ϵ_t is standard normal.

Task 6.5. Simulate an asymmetric SETAR which is defined as

$$\begin{aligned} y_t &= y_{t-1} + \epsilon_t \text{ if } |y_{t-1}| < 5 \\ &= 0.6 y_{t-1} + \epsilon_t \text{ if } y_{t-1} \geq 5 \\ &= 0.95 y_{t-1} + \epsilon_t \text{ otherwise} \end{aligned}$$

where ϵ_t is standard normal.

Task 6.6. Use a ternary operation to assign the string 'positive' and 'negative' based on a simulated standard normal.

Task 6.7. Write a list comprehension that will compute all powers of 2 between 0 and 20.

Task 6.8. Rewrite the previous using a standard for loop.

Task 6.9. Use a list comprehension to find all search queries longer than 30 characters.

7 Special Arrays

Task 7.1. Function Inputs

- Two types of inputs: positional and keyword
  - Positional: array ([1,2],' int16')
  - Keyword: array ([1,2], dtype='int16') or array(dtype='int16',object=[1,2])
- *args: (see plot?)
- **kwargs allow many keyword arguments (see dict?)
- Default values can be provided (see array?)

Task 7.2. Function Outputs

- Functions can return 0 or more values
- Multiple can be either returned as a tuple or split
  - out = broadcast_arrays(x,y)
  - xb, yb = broadcast_arrays(x,y)
  - out = scipy.linalg.svd(z) (How do you get access to this function?)
  - u, s, v = scipy.linalg.svd(z)
- Requesting more produces error

Task 7.3. Generating 1-dimensional arrays

- arange
- linspace and logspace
- r_ (slice-like)
- c_ (slice-like)

Task 7.4. Generating higher dimensional arrays

- zeros, ones and empty
  - like versions
- eye

8 Arrays Functions

Task 8.1. sum and product, cumsum and cumprod

- Use the functions on an array created using random.random((10,2))
- Importance of axis

Task 8.2. Exponentiation: exp, log and log10

- Why is there no inverse function for log10?

Task 8.3. Absolute values: abs and sign.

- Use sign to compute abs of a matrix with both positive and negative values.

Task 8.4. Differences: diff.

- Use diff to compute first differences across columns and rows of a 100 by 2 array (use arange and reshape). How can second differences be computed? Verify your answers using slices.

Task 8.5. Rounding

- around
- floor and ceil
- What is the difference between these three functions when applied to x = random_sample(10)?

Task 8.6. Set functions

- unique
- in1d
- intersect1d
- union1d
- setdiff1d
- Generate x = random_integers(0,30,20), y = random_integers(0,30,20) to explore these functions.

Task 8.7. Sorting and extreme values

- Generate a vector using standard_normal(10) and then use sort and .sort() on the vector. What is the difference?
- amax, amin, .max() and .min()

9 Special Values and Numeric Limits

Task 9.1. inf and nan

- nan-functions: nansum, nanmax, nanmin
  - NumPy 1.8: nanmean, nanvar, nanstd
- Generate x = random_sample(20), x[x<0] = nan and use the nan functions.
- isnan is technically logical, but useful with sum
- How can nanvar be replicated in NumPy 1.7?

Task 9.2. Limits of numeric precision

- Maximum and minimum floating point values
  - finfo(float).max and finfo(float).tiny
  - What happens if you multiply finfo(float).max by a number larger than 1?
- Numeric ϵ
  - finfo(float).eps
  - What about finfo(float).tiny by ϵ? By a number smaller than ϵ?
- Best practices

Task 9.3. Determine which of the following are the same

- 10 and $10+\epsilon$
- 10 and $10+10\epsilon$
- 0 and $\epsilon/2$
- 0 and `tiny`
- 0 and `tiny * eps`
- 0 and `tiny * eps / 2.0`

Task 9.4. Answer the following.

- What is the smallest positive number different from 0.0?
- What is the smallest number that is the same as -1.0?
- What is the largest number that is less than and distinct from ϵ?
- Are $+\infty$ and $-\infty$ the same?

10 Reading and Saving Data using pandas

Task 10.1. Reading data using pandas

- Read the real GDP data from FRED (CSV)
- Read the World Bank debt data (Excel)
- Read the Fama-French data (Text)
- Read the search data from my site (Text)

Task 10.2. Save each `DataFrame` to a separate h5 file.

Task 10.3. Create a random array and use pandas to save an arbitrary NumPy array by wrapping it in a `DataFrame.`

Task 10.4. Save all of the previously created variables using `HDFStore.`

Task 10.5. DataFrames are dictionary-like for accessing values

- See DataFrame.head and DataFrame.tail
- See DataFrame.keys

Task 10.6. Transform the numeric values in the imported data to a standard NumPy array

- asarray
- values

11 Logical Operations and Assignment

Task 11.1. Generate an array between -5 and 5 (inclusive) with 11 elements. Use logical selection to find elements where:

- $|x| \leq 2$
- $x < 0$
- $x \geq 0 \cup x \leq 3$

Task 11.2. Generate a square array using reshape(arange(100),(10,10)) and use ix_ to select all columns and rows where the column sum is above 500 and the row sum is less than 500.

Task 11.3. Using the same matrix, select columns 1,3,4 and 9 and rows 0,2 and 3 using ix_.

Task 11.4. Select the same elements using broadcastable arrays.

Task 11.5. Select the same elements without broadcasting.

Task 11.6. Generate a vector of 0s and 1s where $x = 0$ with probability 50%. Generate a corresponding vector which is $N(0,1)$ when $x = 0$ and $N(2,1)$ when $x = 1$.

Task 11.7. Verify that for any NumPy vector, all elements are either finite, infinite or nan using isfinite , isinf and isnan.

Task 11.8. Let x=arange(10), what is the value of

- x[0]
- x [[0]]
- x[array ([0])]
- x[[True,False,True]]
- x[array ([True,False,True])]
  Why?

12 Custom Functions

Task 12.1. Write a function that will compute the L2 norm for 2 vectors.

Task 12.2. Improve the previous function so that it can compute the L_p norm defined as

$$L_p(x, y) = \left(\sum_{i=1}^{k} |x_i - y_i|^p \right)^{1/p} \quad \text{for } p \geq 1$$

where the default $p = 2$.

Task 12.3. Write a function which can take any number of positional or keyword arguments and will print their contents.

Task 12.4. Write a simple save function based on HDFStore that will take a mandatory filename and an arbitrary list of keyword arguments where each contains a pandas DataFrame sand saves to a compressed h5 file. Return True for success. For example, saveh5('myfile.h5',x=X,gdp=GDP) would save the data in X and GDP to a store using the keys x and gdp, respectively.

Task 12.5. Return to your L_p function and write a good docstring. See the NumPy guidelines at https://github.com/numpy/numpy/blob/master/doc/HOWTO_DOCUMENT.rst.txt

Task 12.6. Call your L_p function from another python file.

13 Linear Algebra Functions

Task 13.1. Initialize

$$x = \begin{bmatrix} 1 & 0.9 \\ 0.3 & 2 \end{bmatrix}$$

and compute the determinant, trace, inverse, inverse, pseudo-inverse and eigenvalues using the linear algebra functions as well as directly using manual element selection and/or matrix multiplication.

- Are the inverse and pseudo-inverse different?

Task 13.2. Determine the complete location of each linear algebra function using ?

14 Random Number Generation

Task 14.1. Use numpy.random to simulate data from:

- $N(8, 20)$
- χ^2_5
- $f_{3,100}$
- Standardized t_4 where the standardization makes the variance 1 ($\mathrm{V}[X] = \nu/\nu-2$ when $X \sim t_\nu$).

Task 14.2. Repeat the previous exercise using scipy.stats.

Task 14.3. Repeat the previous exercise using frozen generators.

Task 14.4. Use numpy.random.RandomState to produce the same sequence of random numbers.

15 Numerical Optimization

Task 15.1. Use fmin_bfgs to minimize the following functions:

- $x^2 - 1$
- $a\,x^2 + b\,x + c$

Task 15.2. Repeat the previous with fmin

Task 15.3. Maximize the Normal log-likelihood,

$$\sum_{i=1}^{n} -\frac{1}{2}\left(\ln 2\pi + \ln \sigma^2 + \frac{(y_i - \mu)^2}{\sigma^2}\right)$$

using re-parameterization.

Task 15.4. Repeat the previous using constrained optimization and fmin_slsqp.

Task 15.5. Download data for the S&P 500 from FRED using

import pandaspandas.read_csv('http://ichart.finance.yahoo.com/table.csv?s=%5EGSPC&ignore=.csv')ret = 100 ∗ sp500['Ad

maximize the likelihood of a standardized Student's T

$$f(x;\nu,\mu,\sigma_t^2) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)}\,\frac{1}{\sqrt{\pi(\nu-2)}}\,\frac{1}{\sigma_t}\,\frac{1}{\left(1+\frac{(x-\mu)^2}{\sigma_t^2(\nu-2)}\right)^{\frac{\nu+1}{2}}}$$

where μ is the mean, σ^2 is the variance, $\nu > 2$ is the degree of freedom and where $\Gamma()$ is known as the gamma function

Task 15.6. Repeat the previous exercise using weekly (5 day) and monthly (22 day) returns. Hint: Use pct_change(5) and notnull to construct the weekly data.

16 Dates and Times

Task 16.1. Construct a datetime for today.

Task 16.2. Get today's date and time using a function.

Task 16.3. Compute tomorrow's date.

Task 16.4. Convert string dates of the following forms to date and datetime:

- yyyymmdd
- mmm-dd-yyyy
- dd/mm/yy hh:mm:ss

Task 16.5. Repeat the previous exercise for numpy.datetime64.

17 Data Management with pandas

Note: All raw data files are available on the course website.

Importing Data

Task 17.1. Use read_csv to read the S&P 500 file.

Task 17.2. Use the url
http://ichart.finance.yahoo.com/table.csv?s=%5EGSPC&ignore=.csv
and read_csv to directly read in data from the web.

Task 17.3. Add index_col and parse_dates to improve the DataFrame produced.

Task 17.4. Load the data into new variables using both read_hdf and read_pickle.

Task 17.5. Read in the Real GDP csv file using DATE as the index column.

Task 17.6. Read in the Real GDP Excel file using read_excel with DATE as the index column. You will need to use the keyword argument skiprows.

Task 17.7. Read in the Real GDP text file using read_table with DATE as the index column. You will need to use the keyword arguments skiprows and sep=r'\s' (which means any space).

Task 17.8. Read-in the GDP potential excel file in a similar manner.

Task 17.9. Use read_stata to read the file TBL5-1.dta.

Task 17.10. Use na_values with read_csv to read CRSP data.

s

Exporting Data

Task 17.11. Export Real GDP to the following formats:

- CSV
- Excel
- Tab seperated
- Latex
- HDF
- HDF with compression
- pickle
- Stata - Note: The index much be converted to "native" and not a datetime when using Stata.

and read the files back in.

Series

Task 17.12. Explore the Series object, which is the pandas version of a 1-dimension array.

- head()
- .name to assign a name, or create using the keyword argument name=
- .index to assign an index, or create using the keyword argument index=
- .index.name to assign name to an index
- Series is array-like, and work wit NumPy functions (e.g. np.log)
- Create a Series for the output gap, using real GDP and real potential GDP
  - How does math work on a Series?
- dropna() to remove NaNs.
- Series can be created from arrays (or other lists) or dictionaries.
- Slicing a Series can be done
  - Numerically, scalar or slice
  - Using index labels, scalar or slice of these as well

DataFrames

Task 17.13. Explore the DataFrame object

- .head(), tail (), .info() and .dtypes()
- Use .column to set and get column names
- Add a series using dictionary notation df['new_series']=series
  - Left join
- Join two or more series using Add a series using dictionary notation pd.concat((a,b))
  - Outer join
- Use .dropna() to remove NaNs
- Use .reindex to alter indices
- Extract a series using
  - dictionary notation
  - .series attribute notation
- Extract a DataFrame from a DataFrame using a list of column names
- Reorder series using a list of column names
- Extracting rows using .ix[*slice*]
- Extracting rows and columns using .ix[*slice,columnName*]

  - Can also use pure numeric slices
  - **Note**: Same caveats as scalar selection and slice selection
- Use xs to extract rows using labels
- Logical indexing works on rows
- Deleting columns
  - del df['series']
  - series = df.pop('series')
  - df.drop('series', axis=1)
- Extract the NumPy array using .values
- Get and set the index using .index
- Create a DataFrame from an array or nested list
  - Keyword arguments: index, columns
  - Can also set later using .index and .columns
    - Note: must have correct number of values
- Create a DataFrame from a dict of series
- Use .copy to copy a DataFrame
- drop to remove rows
  - Can also slice rows to keep
- Sorting
  - .sort_index()
  - .sort(*columnName*) (or list of column names)
    - Keyword arguments: inplace=True
- pivot a DataFrame to get a better representationvalues

Multilevel Indices

- Construct a multilevel index for the world bank data using country code and year
  - Use ix[outerIndex] or xs(outerIndex) to access "groups"
  - Use xs(innerIndex, level=1) to access inner index "groups"
  - Direct access to specific elements with ix[outer, inner]
  - .swaplevel to alter order for easier access
  - . sortlevel to sort a specific level

Missing Values

Task 17.14. Fill missing values using:

- fillna, ffill, bfill
- interpolate
- Can always drop using dropna

18 Plotting

Task 18.1. Import the data from course_data.h5 using

```
import pandas as pd

hdf_store = pd.HDFStore('course_data.h5', 'r')
_g = globals()
for key in hdf_store.keys():
    variable_name = key.split('/')[-1]
    print('Loading ' + variable_name)
    _g[variable_name] = hdf_store[key

hdf_store.close()
del _g
```

Task 18.2. Plot the output gap and year-over-year industrial production growth.

Task 18.3. Plot real GDP growth as a filled line.

Task 18.4. Split GDP growth in to positive and negative parts, where in the negative series positive values are assigned the value 0 (similar for positive). Fill plot the positive values as green and the negative values as red.

Task 18.5. Use the formatting options to plot the output gap in red with stars, and industrial production in purple with circles.

Task 18.6. Download FTSE and S&P 500 data from Yahoo! and merge it to form Thursday-to-Thursday returns. Scatter these returns.

Task 18.7. Download recent world bank data on real GDP growth. Produce a bar chart with strong growth (>3%), weak growth (0-3%), weak recession (-3–0%) and strong recession.

Task 18.8. Produce the previous plot as a horizontal bar chart.

Task 18.9. Plot a histogram of the FTSE 100 weekly returns.

Task 18.10. Plot GDP as a green fill. Then, holding the plot, plot potential GDP.

Task 18.11. In the previous plot, add the title "Real GDP and Potential GDP ($bil)" and the legend for the two lines.

Task 18.12. A bivariate normal has a pdf of

$$f(\mathbf{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = (2\pi)^{-1}\left|\boldsymbol{\Sigma}\right|^{-1/2}\exp\left(-1/2\mathbf{x}'\boldsymbol{\Sigma}^{-1}\mathbf{x}\right)$$

Write a function that will compute this, and evaluate it on a grid from -3 to 3 (for both variables, separately) with a resolution of .05 where

$$\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 1/2 \\ 1/2 & 1 \end{bmatrix}.$$

Produce a contour plot of the output.

Task 18.13. Produce surface and wireframe plots of the same.

Task 18.14. Plot real GDP and transform the y-axis to have a log scale.

Task 18.15. Label the x- and y-axes in the previous plot.

Task 18.16. Export the figure to png, eps, pdf and svg. (savefig)

Task 18.17. Plot the real GDP series (without dates) and use xticks to set the tick locations and dates. You will need to manually format the dates to be strings.

Task 18.18. Produce a plot with both the output gap as well as real GDP and potential GDP (twiny)

Task 18.19. Use figure to get an existing figure, and axis to get its axis.

Problem 18.20. Write a function named multi_bar that will provide a barchart from either a 1-dimensional array or a 2-dimensional array, where the 1D array is treated as n by 1 and the 2-dimensional array is n by k where n is the number of bar locations and k is the number of bars at one locations.

def multi_bar(heights, left=None, width=None, colors=None)where left is the left side of the first bar (n vector), width is a scalar, n vector or n by k array, and colors is a tuple of colors (e.g. ('b','r','g')). If left is None, then np.arange(n) should be used. If width is None, then 1.0/(k+1) should be used. If colors is None, then

('b','g','r','c','m','y','k')

should be used (and should be cycled through if $k > 7$).

19 Fancy Indexing

Task 19.1. Using the Real GDP data from the previous exercise, select the periods with severe recessions (growth < -1%).

Task 19.2. Using the merged FTSE and S&P 500 data, compute the correlation when both are negative and both are positive.

Task 19.3. Simulate a 10 by 10 standard normal matrix. Select all rows and columns where both the row and column sum are negative.

20 Performance Considerations

Task 20.1. Write your own dot function that will multiply two matrices using only loops. Use %timeit to measure the performance on 100 by 100 matrices.

Task 20.2. Replace the inner loop with a NumPy dot. Remeasure performance.

Task 20.3. Measure the performance of NumPy's native dot.

Task 20.4. Write a function to simulate a stationary ECM given by

$$\Delta \mathbf{y}_t = \Phi_0 + \alpha\beta' \mathbf{y}_t + \sum_{i=1}^{P} \Phi_i \Delta \mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t$$

and use %timeit to measure the performance on on a $T = 500$, $k = 2$, $P = 2$ where $\alpha = [.2, -.1]'$ and $\beta = [1, -1]$. You can freely choose Φ_j as long as $\Delta \mathbf{y}_t$ is stationary.

Task 20.5. Use autojit to accelerate the ECM simulation. Measure the performance with %timeit.

Task 20.6. Measure the performance of the previous problem if the output array is pre-allocated.

21 Object oriented Programming

Task 21.1. Make a class to implement a Kalman Filter and Smoother