Introduction to Data Science in Python

Nicholas P. Ross

November 14, 2016

Abstract

Here are my (NPR's) notes on the "Introduction to Data Science in Python" Coursera course from the University of Michigan that I'm taking in November 2016. The URL for that course is `https://www.coursera.org/learn/python-data-analysis/home/welcome`. The URL for these notes is: `https://github.com/d80b2t/Research_Notes/tree/master/Python`

Contents

1 Week 1: Python Fudamentals

1.1 Introduction to Specialization

Kinda a preamble!
General Course Outline (4 modules)
1. General Python Basics
2. The *pandas* Toolkit
3. Advanced Querying and Manipulation in *pandas*
4. Basic Statistical Analysis with *numpy* and *scipy*, and project.

1.2 Syllabus

`https://www.coursera.org/learn/python-data-analysis/supplement/68grE/syllabus`.

If you're having problems, here are a couple of great places to go for help:

- 1. If the problem is with the Coursera platform such as verification on assignments, in video quiz problems, or the Jupyter Notebooks, please check out the Coursera Learner Support Forums.

- 2. If the problem deals with understanding the assignment or how to use the Jupyter Notebooks, please read our Jupyter Notebook FAQ page in the course resources.

- 3. If you have questions with the content of the course, or questions about programming in python or with the toolkits described, you can contact your peers and the course instructors in the discussion forums, or go to Stack Overflow.

1.3 Data Science

`http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram`

David Donoho, Professor of Statistics in Stanford., "50 Years of Data Science". 1. Data Exploration and Preparation.
2. Data Representation and Transformation.
3. Computing with Data.
4. Data Modeling.
5. Data Visualization and Presentation.
6. Science about Data Science.

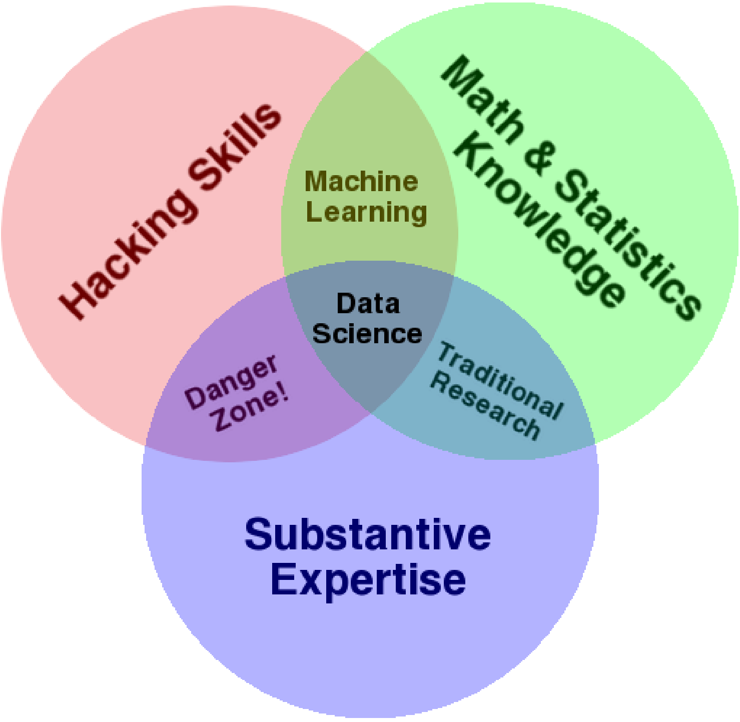

Figure 1: Drew Conway's Venn Diagram.

1.4 The Coursera Jupyter Notebook System

All pretty standard, straighforward.

1.5 Python Functions

Of course, Python has traditional software structures like functions. Here's an example, refactoring that previous code into a function. You'll see the def statement indicates that we're writing a function. Then each line that is part of the function needs to be indented with a tab character or a couple of spaces.

```
def add_numbers(x, y):
    return x + y

add_numbers(1, 2)
```

Okay, functions are great but they're a bit different than you might find in other languages and here are some of subtleties involved. First, since there's no typing, you don't have to set your return type. Second, you don't have to use a return statement at all actually. There's a special value called None that's returned. None is similar to null in Java and represents the absence of value. Third, in Python, you can have default values for parameters. Here's an example.

```
def add_numbers(x,y,z=None):
    if (z==None):
        return x+y
    else:
        return x+y+z

print(add_numbers(1, 2))
print(add_numbers(1, 2, 3))
```

In this example, we can rewrite the add numbers function to take three parameters, but we could set the last parameter to be None by default. This means that you can call add numbers with just two values or with three, and you don't have to rewrite the function signature to overload it.

```
def do_math(a, b, kind='add'):
  if (kind=='add'):
    return a+b
  else:
    return a-b
```

1.6 Python Types and Sequences

1.7 Python More on Strings

1.8 Python Demonstration: Reading and Writing CSV files

1.9 Python Dates and Times

1.10 Advanced Python Objects map()

1.11 Advanced Python Lambda and List Comprehensions

1.12 Advanced Python Demonstration: The Numerical Python Library (Numpy)

2 References and Bibliography