DSC 201: Data Analysis & Visualization

Python and Notebooks

Dr. David Koop
“Computer-based visualization systems provide visual representations of datasets designed to help people carry out tasks more effectively.”

— T. Munzner
Visual Representations

\[ y_1 \]

\[ y_2 \]

\[ y_3 \]

\[ y_4 \]
Tasks & Visual Pop-out

Visual Encoding

• How do we encode data visually?
  - **Marks** are the basic graphical elements in a visualization
  - **Channels** are ways to control the appearance of the marks

• Marks classified by dimensionality:

  - **Points**
  - **Lines**
  - **Areas**

• Also can have surfaces, volumes

• Think of marks as a mathematical definition, or if familiar with tools like Adobe Illustrator or Inkscape, the path & point definitions
Channels by Effectiveness

Channels: Expressiveness Types and Effectiveness Ranks

**Magnitude Channels: Ordered Attributes**
- Position on common scale
- Position on unaligned scale
- Length (1D size)
- Tilt/angle
- Area (2D size)
- Depth (3D position)
- Color luminance
- Color saturation
- Curvature
- Volume (3D size)

**Identity Channels: Categorical Attributes**
- Spatial region
- Color hue
- Motion
- Shape

[Muñzner (ill. Maguire), 2014]
Color and Wavelength

But rather, a combination of wavelengths and energy.

[Image: Graph showing relative energy density vs. Wavelength (nm) with peaks labeled yellow and brown.]
Color Blindness

Normal

Prostanopia

Deuteranopia

Tritanopia

[via M. Meyer]
Colormap

- A colormap specifies a **mapping** between data values and colors
- Colormap should follow the expressiveness principle
- Types of colormaps:

  **Binary**
  - y
  - n

  **Diverging**
  - -1
  - 0
  - +1

  **Categorical**
  - T
  - F
  - A

  **Sequential**
  - 3
  - 2
  - 1

[Munzner (ill. Maguire), 2014]
Artifacts from Rainbow Colormaps

[Borland & Taylor, 2007]
Textbook Available

- eBook now available
- Paperback later this month
- I will be referring to this text a lot during the rest of the course
Assignment 2

• www.cis.umassd.edu/~dkoop/dsc201-2017fa/assignment2.html
• Python and Jupyter Notebook
• Dataset of MA, CT, RI towns
• Data exploration
  - Find max, min
  - Count
  - String processing
• Due next Wednesday
• Start now!
Chicago Food Inspections Exploration
Chicago Food Inspections

- Do not focus on code syntax!
- What questions did we analyze?
Chicago Food Inspections

• Do not focus on code syntax!
• What **questions** did we analyze?
• Sample answers:
  - How many inspections are there?
  - What types of ratings do facilities get?
  - Which restaurant chains fail inspections most?
  - What address has the most inspection failures?
  - What is unique about that address?
  - How often is a facility checked?
  - Which violation is most common?
  - Where are failures most common?
Chicago Food Inspections

• What tasks did we do?
Chicago Food Inspections

• What **tasks** did we do?

• Sample answers:
  - Reading data
  - Organizing data
  - Extracting unique values
  - Counting occurrences
  - Cleaning names
  - Searching data
  - Transforming data (string -> list)
  - Visualizing data
Python

• Started in December 1989 by Guido van Rossum
• “Python has surpassed Java as the top language used to introduce U.S. students to programming…” (ComputerWorld, 2014)
• Python and R are the two top languages for data science
• High-level, interpreted language
• Supports multiple paradigms (OOP, procedural, imperative)
• Help programmers write readable code
• Use less code to do more
• Lots of libraries for python
  - Designed to be extensible
  - Easy to wrap code from other languages like C/C++
• Open-source with a large, passionate community
Python Compared to Java

• Dynamic Typing
  - A variable does not have a fixed type
  - Example: \( a = 1; \ a = \text{"abc"} \)

• Indentation
  - Braces define blocks in Java, good style is to indent but not required
  - Indentation is critical in Python
  - Example:
    
    ```
    z = 20
    if x > 0:
      if y > 0:
        z = 100
      else:
        z = 10
    ```
Learning Python Resources

- https://software-carpentry.org/lessons.html
- http://www.pythontutor.com
- http://www.python-course.eu
- http://thepythonguru.com
- https://en.wikibooks.org/wiki/A_Beginner%27s_Python_Tutorial
- https://learnpythononthehardway.org
- learnpython.org
Jupyter Notebook

- Display rich representations and text
- Uses Web technology
- Cell-based
- Built-in editor
- GitHub displays notebooks
Jupyter Notebook

- An interactive programming environment
- Runs in your web browser…
- … but does not require an internet connection
- Displays results (even interactive maps) inline
- Originally designed for Python
- Supports other languages, too
- You decide how to divide code into executable cells
- Shift+Enter to execute a cell
Obtaining Python (and Jupyter Notebook)

- www.anaconda.com/download/
- Anaconda has Jupyter Notebook
- Use Python 3.6 version (not 2.7)
- Anaconda Navigator
  - GUI application for managing Python environment
  - Can install packages
  - Can start Jupyter Notebook
- Can also use the shell to do this:
  - $ jupyter-notebook
  - $ conda install <pkg_name>
Jupyter Notebook

• Starts with a directory view
• Create new notebooks using "New" menu on the right
  - Originally has name "Untitled"
  - Click on "Untitled" to change the name (do this!)
• Save a notebook using the command under the File menu
• Shutting down the notebook requires quitting the kernel
  - Web browser is interface to display code and results
  - Kernel actually runs the code: usually see messages in a console/terminal window
  - Ctrl+C in this window ask you if you want to shut down the kernel
Jupyter Notebook

• Open a notebook by going back to the main screen and clicking on it like you would in a desktop view
• Past results are displayed—does not mean they are loaded in memory
• Use "Run All" or "Run All Above" to re-execute past work
  - If you shut down the kernel, all of the data and variables you defined need to be redefined (so you need to re-run all)
  - **Watch Out—Order Matters:** If you went back and re-executed cells in a different order than they are shown, doing "Run All" may not produce the same results!
• Edit mode (green) versus Command mode (blue == **Be Careful**)

D. Koop, DSC 201, Fall 2017
**Jupyter Notebook**

- Can write code or plain text (can be styled Markdown)
  - Choose the type of cell using the dropdown menu
- Cells break up your code, but all data is **global**
  - Defining a variable $a$ in one cell means that variable is accessible in any other cell
  - This includes cells above the cell $a$ was defined in!
- Remember **Shift+Enter** to execute
- Enter just adds a new line
- Use `?<function_name>` for help
- Use Tab for **auto-complete** or suggestions
- Tab also indents, and Shift+Tab unindents
Jupyter Notebook Documentation

- Notebook Quick Start Guide
- Notebook Basics
Advanced: Python 2 and 3

- [https://docs.python.org/3/whatsnew/3.0.html](https://docs.python.org/3/whatsnew/3.0.html)
- Key Differences:
  - *print as a function*: `print "Hello"` vs. `print("Hello")`
  - Views and iterators instead of lists
  - Integer division: `5/2 = 2.5, 5//2 = 2`
  - Unicode as standard
  - String formatting:
    - Py2: "Hello %s. You are %d years old" % (name, age)
    - Py3: "Hello {}. You are {} years old".format(name, age)
    - Py3.6: f"Hello {name}. You are {age} years old"
Print function

- `print("Hello World")`
- Can also print variables:
  ```python
  name = "Jane"
  print("Hello,", name)
  ```
Python Variables and Types

• No type declaration necessary
• Variables are names, not memory locations
  
  ```
  a = 0
  a = "abc"
  a = 3.14159
  ```

• Don't worry about types, but think about types
• Strings are a type
• Integers are as big as you want them
• Floats can hold large numbers, too (double-precision)
Python Math and String "Math"

- Standard Operators: +, -, *, /, %
- Division "does what you want" (new in v3)
  - \(-5 \div 2 = -2.5\)
  - \(-5 \div\ 2 = 2\) # use // for integer division
- Shortcuts: +=, -=, *=
- No ++, --
- Exponentiation (Power): **
- Order of operations and parentheses:
  - \(-4 - 3 - 1\)
  - \(-4 - (3 - 1)\)
- "abc" + "def"
- "abc" * 3
Python Strings

- Strings can be delimited by single or double quotes
  - "abc" and 'abc' are exactly the same thing
  - Easier use of quotes in strings: "Joe's" or 'He said "Stop!"'

- String concatenation: "abc" + "def"

- Repetition: "abc" * 3

- Special characters: \n \t like Java

- Indexing:
  - a = "abcdef"
  - a[0]

- Slicing: a[1:3]

- Format: name = "Jane"
  - print("Hello, {}".format(name))
Exercise

• Given variables x and y, print the long division answer of x divided by y with the remainder.

• Examples:
  - x = 11, y = 4 should print "2R3"
  - x = 15, y = 2 should print "7R1"