PySnippet: Accelerating Exploratory Data Analysis in Jupyter Notebook through Facilitated Access to Example Code

Alex Watson, Scott Bateman and Suprio Ray
Presenter: Virendra C. Bhavsar

University of New Brunswick, Fredericton, Canada

BigVis 2019, Lisbon, Portugal
Outline

• Motivation

• The problem

• The proposed solution

• Experimental methodology

• Results
Data Science

• “Data science is the study of the ... problems that can be solved with data*” (*using tools and methods...).
  - Rachel Schutt and Cathy O’Neil

• “Data science is an interdisciplinary field about processes and systems to extract knowledge or insights from data in various forms, either structured or unstructured.”  -Wikipedia
Data Science tools

- Increasingly, data science and data analysts employ *interactive environments* as the primary tool in their analysis activities
  - Jupyter Notebook
  - Mathematica
  - RStudio 3
  - MATLAB 4
Jupyter notebook

• An open-source, web-based, interactive data analysis environment/tool

• Allows users to create documents that contain
  - live code
  - equations
  - visualizations
  - narrative text

• Users can share documents online

• Rich interactive presentations
Advanced items for beginners

Here we will briefly cover

- 3D plots
- Animations


Author: Thomas Haslwanter, Date: Feb-2017

```python
In [4]: fig = plt.figure(figsize=(14,6))

    # `ax` is a 3D-aware axis instance, because of the projection='3d' keyword argument to add_subplot
    ax = fig.add_subplot(1, 2, 1, projection='3d')

    p = ax.plot_surface(X, Y, Z, rstride=4, cstride=4, linewidth=0)

    # surface plot with color grading and color bar
    ax = fig.add_subplot(1, 2, 2, projection='3d')
    p = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=plt.cm.viridis, linewidth=0, antialiased=False)
    cb = fig.colorbar(p, shrink=0.5)
```

![3D plot images]
Data Science Ecosystem

- Growing libraries, application and community
- Anaconda Python
  - Anaconda is the most popular Python-based data science platform.
  - Includes more than 1400+ popular data-science packages
Data Science Ecosystem

- Growing libraries, application and community

- Comprehensive R Archive Network (CRAN)
  - Includes more than 15000+ packages
  - Supports data analytics for a wide range of fields such as Finance, Genetics, High Performance Computing, Machine Learning, Medical Imaging, Social Sciences and Spatial Statistics
The Problem

- **Learning Curve** involved when learning new technologies
- Plot 2 *int* arrays *x* and *y* using matplotlib
  
  - *x* = [1, 2, 3, 4]
  - *y* = [3, 4, 8, 6]

```python
import matplotlib.pyplot
import pylab

x = [1, 2, 3, 4]
y = [3, 4, 8, 6]

matplotlib.pyplot.scatter(x, y)
matplotlib.pyplot.show()
```

`matplotlib.pyplot.scatter`
The Problem

- Finding snippets to complete can be time-consuming
  - Requires time browsing through online documentation or searching online repositories

- Integrating found code can also be time-consuming
  - Need be adapted to the current context of existing code (i.e., variables must be renamed, dependencies must be included)

- Beginners may lack the expertise to successfully interpret and transfer code snippets into their solutions
The Solution: PySnippet

- PySnippet aims to reduce the overall development time
  - Providing an automatic, easy-to-access code snippet feature directly in the Jupyter Notebook environment
  - Allows rapid access to code snippets that illustrate how common tasks can be completed
  - Easy integration into an analyst’s current workbook.
Plot 2 int arrays x and y using matplotlib

- \(x = [1,2,3,4]\)
- \(y = [3,4,8,6]\)
System description

• Built into the Jupyter Notebook Environment

• Seamless integration
  • If a user presses TAB, PySnippet parses the current line as a set of keywords
  • If snippets matching the keywords is found, code snippets are added as options in the code completion list
  • When a snippet is highlighted in the list, a small description of the code snippet appears in a pop-up to the right of the list showing the corresponding code snippet
Example scenario

```python
import numpy as np
import matplotlib.pyplot as plt

# Create data
N = 500
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
area = (30 * np.random.rand(N))**2

# Scatter Plot
plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.title('Scatter plot')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```
Evaluation

- Two versions of Jupyter notebook
  - Normal
  - PySnippet

- Independent Variable
  - One independent variable with two levels (System version: normal Jupyter, and Jupyter with

- Dependent Variables
  - Completion Time
  - Google search count
  - Incorrect submission count
Evaluation - participants

- Participants
  - Eight participants were recruited

- Programming experience
  - Seven had programming experience at an undergraduate level
  - Five had programmed in a professional environment
  - One participant had little to no programming experience

- Python experience
  - Four had no or minimal experience in Python and had never worked with Jupyter,
  - The other four had varying degrees of experience with Python
Experimental Methodology

Steps:

1. Questionnaire - Programming Experience

2. Jupyter Notebook Introduction

3. 1 practice trial, then 3 experimental trials (one condition)

4. Complete questionnaire on the condition effectiveness
   - Repeat steps 3 and 4 for the other condition

5. Complete final questionnaire on which condition they preferred
Experimental Procedure – Task example
Results - task completion time

- The mean task completion time for PySnippet was 30.5% less than that of normal
The mean number of Google searches for each task using PySnippet was 92.8% less than that with normal.
Results - incorrect submissions

- The mean number of incorrect submissions was 1.687 for PySnippet and 1.875 for normal

- There was substantial variation across participants

![Bar chart showing the number of incorrect submissions for each participant comparing average normal and average PySnippet submissions.](chart.png)
Post-experiment opinion survey

- Seven out of the eight participants preferred PySnippet over *normal*
- One of the participants was indifferent
- None preferred *normal*
- Some users expressed frustration on having to go back and search online for snippets instead of just using PySnippet
Conclusion

- PySnippet: an automatic code completion engine in Jupyter Notebook – takes keywords provided by users

- Reduce overall development time by having to spend little or no time searching online for common code snippets

- Performed well on the limited set of tasks we evaluated