

Paper 4732-2020

Using Jupyter to Boost Your Data Science Workflow

Hunter Glanz, Cal Poly, San Luis Obispo, CA

ABSTRACT

From state-of-the-art research to routine analytics, the Jupyter Notebook offers an unprecedented reporting medium. Historically, tables, graphics, and other types of output had to be created separately and then integrated into a report piece by piece, amidst the drafting of text. The Jupyter Notebook interface enables you to create code cells and markdown cells in any arrangement. Markdown cells allow all typical formatting. Code cells can run code in the document. As a result, report creation happens naturally and in a completely reproducible way. Handing a colleague a Jupyter Notebook file to be re-run or revised is much easier and simpler for them than passing along, at a minimum, two files: one for the code and one for the text. Traditional reports become dynamic documents that include both text and living SAS®, R, Python or other code that is run during document creation. With Jupyter, you have the power to create these computational narratives and much more!

INTRODUCTION

In the past, scientific research and statistical analyses took place almost exclusively within particular software packages like SAS, Python, R or some other domain-specific program. A single project usually included multiple scripts that compartmentalized tasks like data cleaning, data manipulation, data visualization, statistical analysis and interpretation. Whether these pieces were executed separately or within some main, delegating script, they all stood apart from the write-up or narrative that inevitably accompanies such projects. Of course the code throughout should be well documented/commented, but some of these descriptions and explanations often appeared in the write-up as well. Output and graphics needed to be copied or exported in some way in order to integrate them into the project write-up. In the end, the report reads well and looks nice, but to fully share your project with someone there were numerous files to consolidate and send: code scripts, image files, data files, the codebook for the data, and the project write-up itself. The whole ordeal almost required a separate file with instructions on how to navigate all of these project materials!

As of September 1, 2016 the *Journal of the American Statistical Association: Applications and Case Studies* requires code and data as a minimum standard for **reproducibility** of statistical scientific research [1]. The concept and goal of **reproducibility** seems like it should have always been implicit in all analyses and research, but only in recent years has its explicit popularity exploded. Courses on sites like Coursera emphasize adhering to this principle, and now the American Statistical Association will tangibly require it as part of their publication process. This all means authors are now required to submit collections of materials similar to those described above: possibly multiple code scripts, data files, and the article itself. This process can seem like a hassle and might even increase the potential for errors and problems with more materials to keep track of.

The Jupyter Notebook alleviates the obligation to navigate all of these files by allowing the code, output, graphics, codebook for the data, and narrative text to exist within the same file! With the code in the same file as the text, the possible redundancy between comments in the code and text in the write-up disappears. How does the Jupyter Notebook accomplish all of this?

The Jupyter Notebook is a web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text [2]. The notebook has support for over 40 programming languages, including SAS now. Notebooks are easily shared with others. Code within the notebook can produce rich output such as images, videos, LaTeX, and JavaScript. Interactive widgets can be used to manipulate and visualize data in real time.

Wrapping all of these utilities into one cohesive tool revolutionizes the way we do data science and statistical computing/communication. The benefits of the Jupyter Notebook shone across arenas such as computing coursework, academic research, and numerous industries.

WHERE TO BEGIN

Learning a new tool can be daunting, especially one that accomplishes so much! Thankfully, Project Jupyter [2] makes it easy to install and use by following the instructions at:

<https://jupyter.org/install>

These instructions only get you started with the Jupyter software and Python (the language it was originally built for). In order to use SAS with Jupyter, you will need to install the SAS kernel for Jupyter. The experts at SAS have made this straightforward as well, by following the instructions at their GitHub page here:

https://github.com/sassoftware/sas_kernel

With these set up you will be on your way in no time at all! For a more accessible trial of the SAS-with-Jupyter environment, be sure to check out SAS University Edition. Users of SAS University Edition likely already know that Jupyter Notebooks (and now JupyterLab) have been an alternative to the SAS Studio interface for some time now. This alternative requires no extra effort! Figure shows the welcome screen for SAS University Edition, containing options to either start the SAS Studio interface or the JupyterLab interface.



Figure 1. Homepage of SAS University Edition. Traditional button to start SAS Studio interface is accompanied by an option to start JupyterLab.

With your venue determined, it's a small step to launch your first Jupyter Notebook and begin working with SAS in one of the most exciting new ways!

JUPYTER NOTEBOOKS

Brian Granger, one of the developers of Project Jupyter, often recounts [3]:

"Computers are good at consuming, producing and processing data. Humans are good at consuming, producing and processing stories. For data to be useful to humans, we need tools for telling stories that involve code and data."

This impetus for the creation of Project Jupyter helps define Jupyter Notebooks as a vehicle for what we now call *computational narratives*. Communication of statistical investigations and analyses supersedes all else, but depends on data and code at its core. Without the story or context, data summarizations and visualizations can be dry and meaningless. The Jupyter Notebook accommodates and unifies all of these things within a single environment.

A typical Jupyter Notebook consists of a series of *cells*, as many as you like. These cells can contain code or markdown text. The user is literally creating a living, dynamic document that appears as a typical write-up would but contains live code that you can run at any time. The cells can re-arranged at will and the code cells can be executed altogether or in any order you like.

Though the Jupyter Notebook is a web application, it is easily installed and used on any personal machine. It can also be deployed on centralized servers for use by many different users either within an organization or a class of students. **Jupyter Notebooks with SAS can now also be used from within SAS University Edition!** (as mentioned in the previous section)

Figure 2 shows the header of the "home" page once you have launched Jupyter from your own personal installation. Figure 3 shows the "home" page of JupyterLab, the interface now offered through SAS University Edition.



Figure 2. Header of "home" page of Jupyter. The image is from within a Google Chrome browser, but other browsers would work fine.

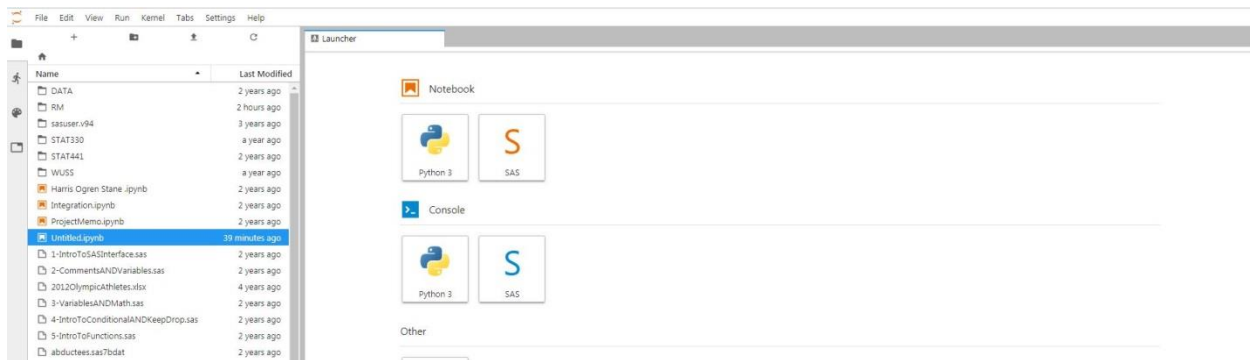


Figure 3. Home screen of JupyterLab through SAS University Edition. File explorer on the left side panel. Notebook launcher on the right main panel.

From here you can navigate throughout your computer or system as you would from within “My Computer” on a PC or even a terminal on Mac/Linux. In fact, the initial installation of Jupyter provides functionality for use as a simple text file editor, a terminal, or the notebook environment (the focus of this paper).

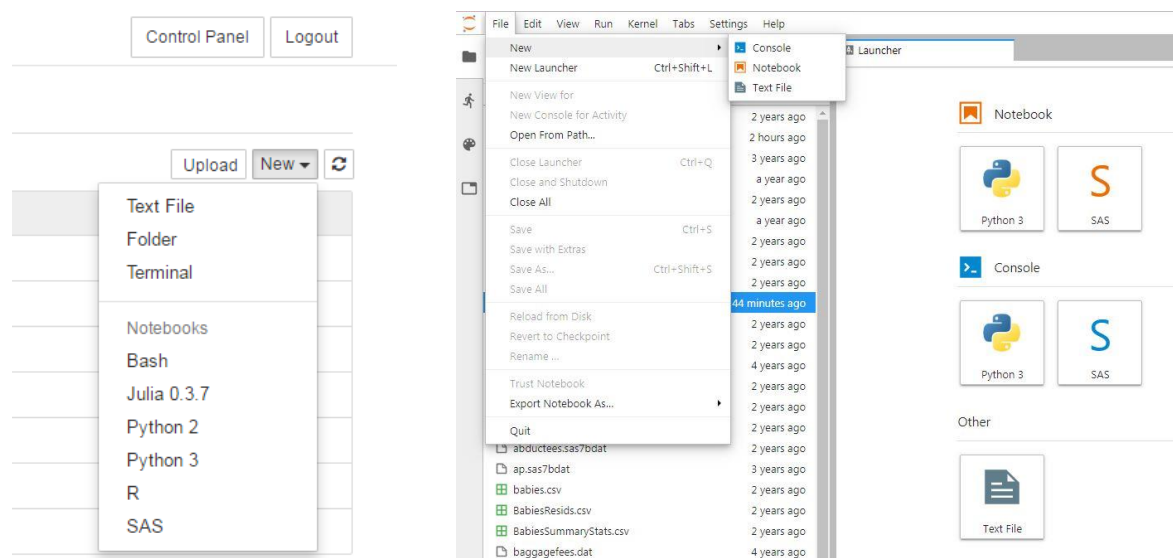


Figure 4. The choice for new applications from within Jupyter (left) or JupyterLab (right). In JupyterLab, one can either use the “File” menu at the top or click the appropriate icon in the main panel.

Figure 4 demonstrates how you might open a new text file, terminal, or notebook within Jupyter. Notice, to open a new notebook you must specify the kernel you would like to use for that notebook. That is, you must choose the base/major programming language that will be in use throughout that notebook. It is possible to use multiple languages within a single notebook, but I will not get into those details here. Based on the image in Figure 2, you can see I can make use of Julia, Python, R, or SAS from within a notebook. ***When working with Jupyter Notebooks within SAS University Edition you currently only have access to a text file editor, folder explorer, and notebooks using SAS or Python (no other languages are available).***

To start a new notebook I need only click on the desired kernel. This will create a new notebook file within my current working directory. The file will then appear under the **Files** tab on your home page (or in the JupyterLab left panel). Because that notebook needs to be able to run code, upon creation it will also show up under the **Running** tab on your home page. Stopping or halting your notebook will not delete or remove it, but just stop the kernel so that your machine no longer spends valuable resources on it. So what does a notebook look like?

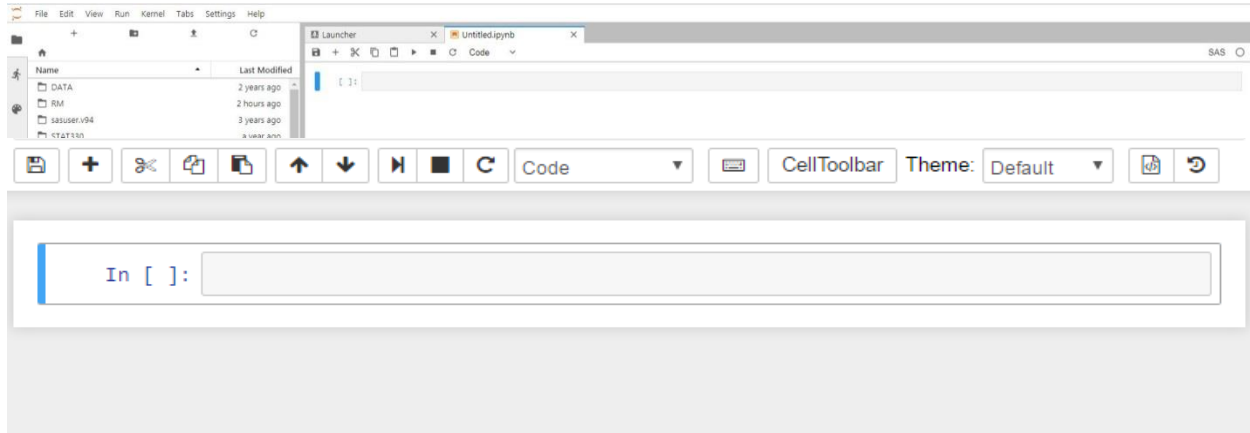


Figure 5. A new Jupyter Notebook with a SAS kernel in Jupyter (top), or JupyterLab (bottom).

Figure 5 depicts a freshly created Jupyter Notebook with a SAS kernel. Jupyter notebooks always display the type of kernel in the top right corner of the page. The name of the file (notebook), currently "Untitled", can be changed by simply double-clicking it at the top.

Jupyter notebooks are made up of a series of *cells*. The flexibility of these cells makes Jupyter the amazing tool that it is. The notebook starts with a single cell, displayed in Figure 5 as the beige box in the middle with "In []:" directly to the left of it. The thin gray box around this cell means that it is selected. The "In []:" notation in addition to the word "Code" at the top of the screen indicate that this is a *code cell*. This means SAS code could be entered into this cell and run. The output would then appear in a cell directly beneath the cell in which the code was run, as seen in Figure 6.

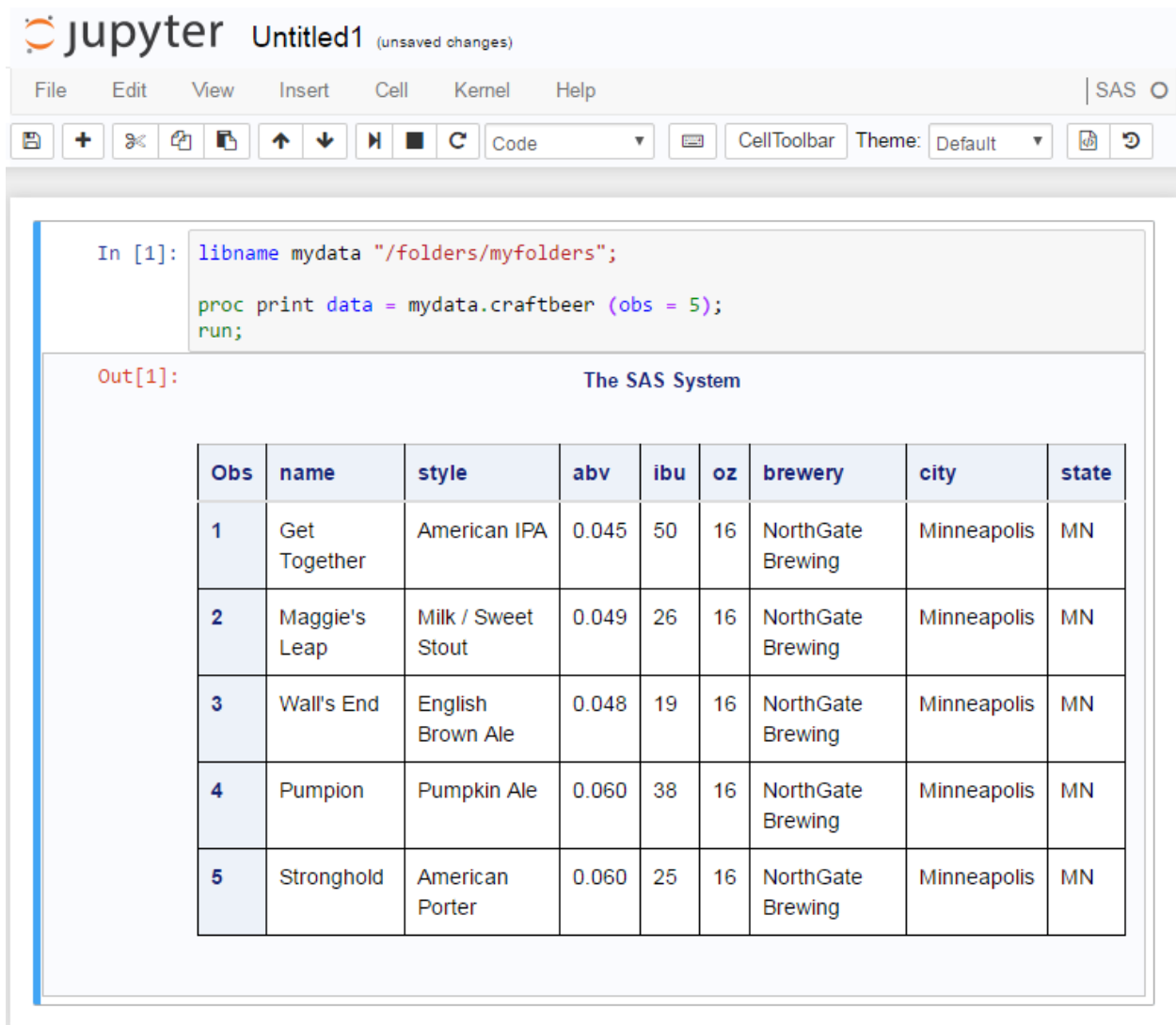


Figure 6. Sample SAS output in a Jupyter Notebook.

Code cells and SAS output cells seem easy enough to use and navigate, but where did the SAS log go? SAS wonderfully integrates two easy-to-use buttons for accessing the SAS log, as seen in Figure 7. Because code cells can be run individually or in groups or all at once, and because Jupyter Notebooks can become quite lengthy, one button accesses the log for the most recently executed cell while the other accesses the log for the entire notebook.

The buttons displayed in Figure 7, for revealing the log, do not exist in the JupyterLab interface. From within JupyterLab, to view the log for SAS code that you have just run you should type `%showLog` or `%showFullLog` in the next cell and then run that cell.

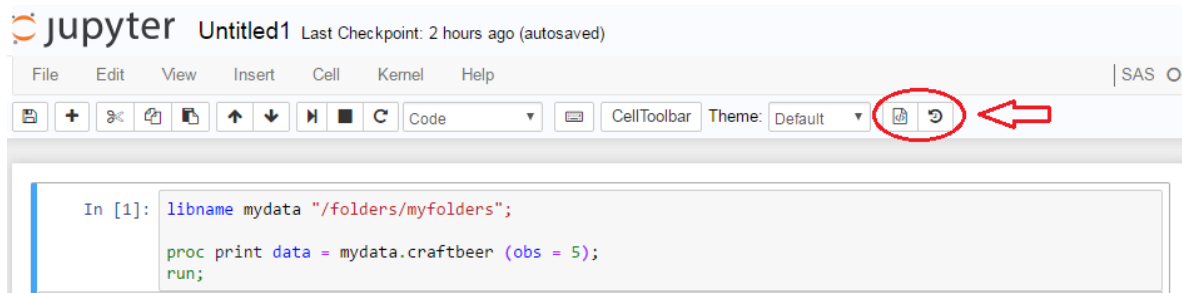


Figure 7. Access to the SAS log in a Jupyter Notebook. The left button shows the SAS log for the last executed cell. The right button shows the complete SAS log for the entire notebook.

Up to this point Jupyter has not provided anything Base SAS does not already provide, **except** that this notebook structure of a series of cells lends itself incredibly well to easily and conveniently running only certain pieces of code or portions of an analysis.

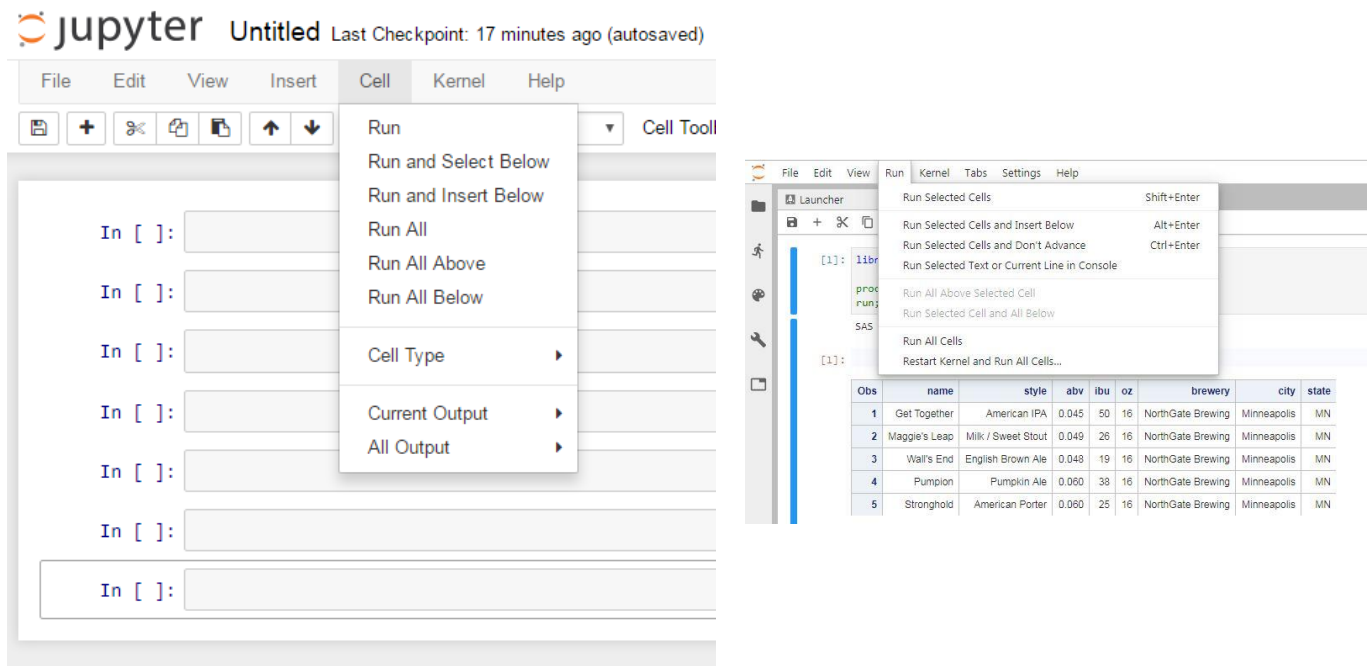


Figure 8. Some of the options and flexibility for running parts of your notebook in Jupyter (left) or JupyterLab (right).

Figure 8 hints at the flexibility Jupyter boasts when it comes to partially running your script or analysis. These pieces, or cells, are much more distinguishable than comment-separated portions of code within a single SAS script.

Jupyter's coup de grace over most, if not all, other tools of this nature is its flexibility in *cell type*. The cells of these notebooks are not restricted to code!

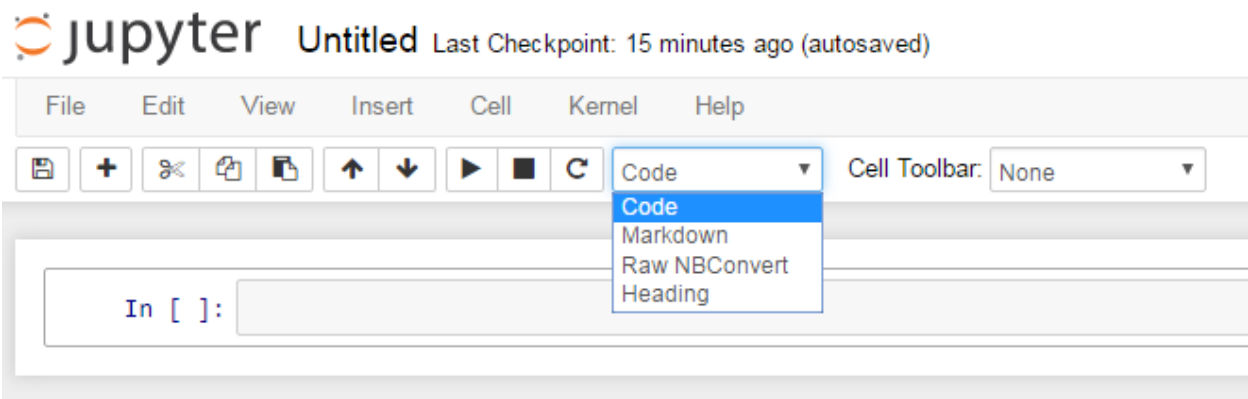


Figure 9. The menu for choosing cell type.

With coding and markdown cells the Jupyter Notebook literally becomes a living, dynamic document! SAS code can be entered and run in one cell, produce output in the next, and be wrapped above and below with text telling the story of the analysis. Jupyter effectively makes the job of report writing seamless and painless.

Notebooks are, indeed, easily shared but we are by no means confined to Jupyter for viewing things.

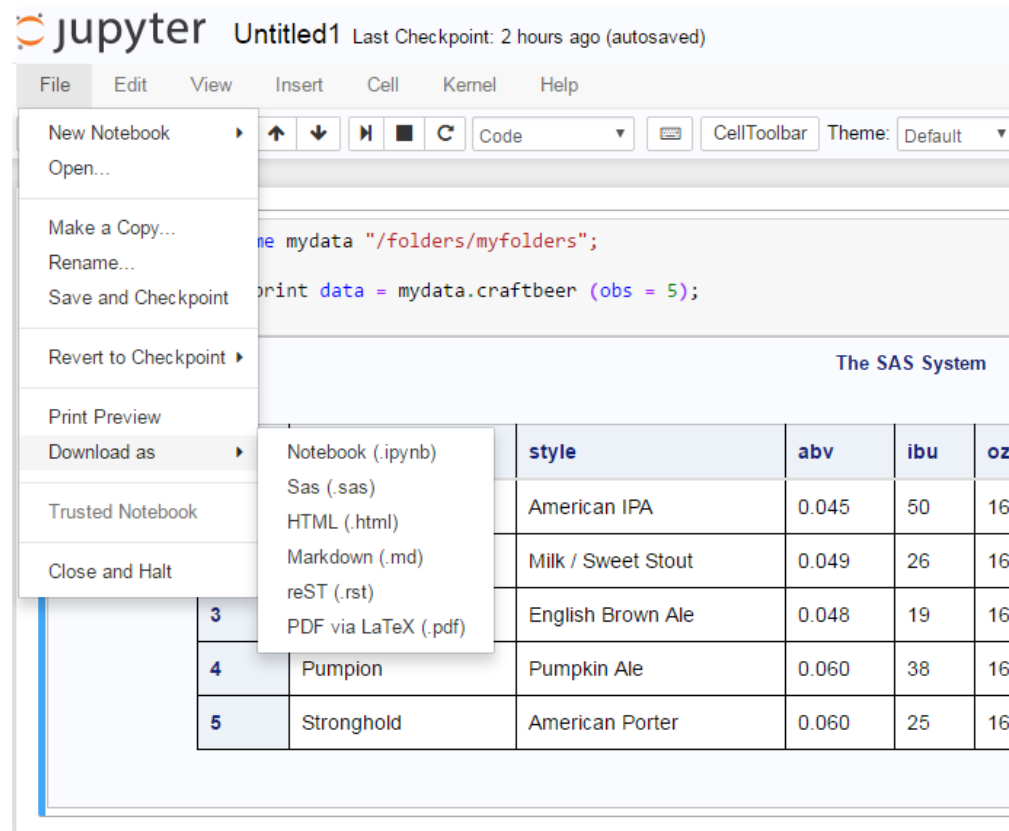



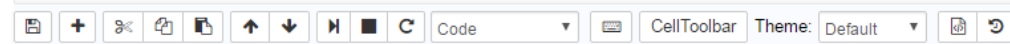
Figure 10. Save types for Jupyter notebooks.

The default extension of a Jupyter Notebook is .ipynb, after the IPython Notebook that preceded Project Jupyter. Figure 10 reveals the many well-used formats that Jupyter notebooks can be downloaded as in addition to .ipynb. Notably, Jupyter notebooks can be converted to HTML or PDF files, which are even more ubiquitous than notebook files...for now. You can even download the code from your entire notebook as a traditional SAS script.

Now it takes a relatively small amount of time to create a coherent, integrated document that is publication quality!

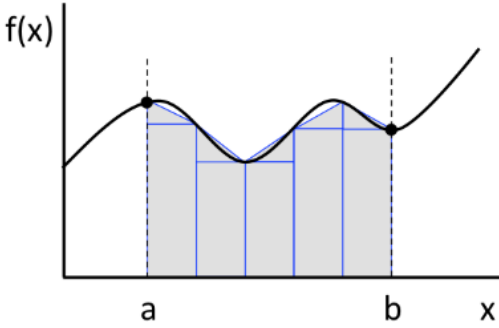

Integration
Last Checkpoint: 4 hours ago (autosaved)

File Edit View Insert Cell Kernel Help
SAS



Trapezoidal Rule

A better approach, which involves very little extra work, is to divide the area into trapezoids rather than rectangles. The area under the trapezoids is a considerably better approximation to the area under the curve, and this approach, though simple, often gives perfectly adequate results.



We can improve the approximation by making the size of the trapezoids smaller. Suppose we divide the interval from a to b into N slices or steps, so that each slice has width $h = (b - a)/N$. Then the right-hand side of the k th slice falls at $a + kh$, and the left-hand side falls at $a + kh - h = a + (k - 1)h$. Thus the area of the trapezoid for this slice is

$$A_k = \frac{1}{2}h[f(a + (k - 1)h) + f(a + kh)]$$

This is the *trapezoidal rule*. It gives us a trapezoidal approximation to the area under one slice of our function.

Now our approximation for the area under the whole curve is the sum of the areas of the trapezoids for all N slices

$$I(a, b) \simeq \sum_{k=1}^N A_k = \frac{1}{2}h \sum_{k=1}^N [f(a + (k - 1)h) + f(a + kh)] = h \left[\frac{1}{2}f(a) + \frac{1}{2}f(b) + \sum_{k=1}^{N-1} f(a + kh) \right]$$

Note the structure of the formula: the quantity inside the square brackets is a sum over values of $f(x)$ measured at equally spaced points in the integration domain, and we take a half of the values at the start and end points but one times the value at all the interior points.

Applying the Trapezoidal rule

Figure 11. A snippet from an example notebook on integration.

Figure 11 displays a small portion of a dynamic document created using a Jupyter Notebook to discuss integration. While the example is a bit pedantic, it demonstrates nicely the full integration of markdown headings, a visualization, accompanying text including LaTeX math notation, and executed but suppressed code.

USING JUPYTER TO DO REPRODUCIBLE DATA SCIENCE

Statistics and Data Science projects often involve an extensive and sometimes intense workflow that can start with data collection. Usually data must be cleaned in some way and then prepared for analysis. Summaries and visualizations supplement both the exploratory data analysis phase and the final analysis itself. This collection of commented scripts must inevitably get cleaned up after the project to make it more easily shared and readable to

others. Even once the scripts are clean, the results of all that work are distinct from scripts; living in their own meticulously drafted report.

The Jupyter Notebook simplifies all of this by assimilating the code, the documentation for said code, the output and graphics, and the project write-up into a single, unified document that ensures reproducibility by allowing live code to be run throughout the document. Granger, again, would say the Jupyter Notebook is “making computational reproducibility enjoyable and minimizing the ‘distance’ between a human user and their code/data through interactivity” [3].

The use of Jupyter Notebooks and support for them continues to grow exponentially to the point where they are becoming a standard environment to work in. Github now renders Jupyter Notebooks and is indexing notebook content, going so far as to track trending notebooks. The Binder project enables Github hosted notebooks to be run by anyone on the internet!

Many online scientific publications now include their data and analysis via Jupyter Notebooks. O’Reilly media publishes books accompanied by Jupyter Notebooks. BuzzFeedNews publishes code and data for all data-backed articles they write, via Jupyter Notebooks. The LA Times is starting to do the same thing!

While Project Jupyter is free to use, it’s gaining popularity in commercial products as well. Companies like IBM, Kaggle, Google Cloud Datalab, DataRobot, dataiku, and many more are building dashboards, platforms and other products based on Jupyter Notebooks.

The academic uses for Jupyter are just as obvious as the industry-based ones! In parallel with the explosion of reproducibility as a standard for data science work is the increasing trend of tool plurality among data scientists. Whether it’s a variety of tools or just a small handful, more and more people who work with data do that work with multiple software packages: SAS, Python, R, Tableau, and many more. **Jupyter’s compatibility with numerous languages, including SAS, Python, and R, is another huge benefit it boasts!**

USING JUPYTER TO REINFORCE SOFT SKILLS IN THE CLASSROOM

Historically, writing reports and presentations involved cobbling together separately created code, output, and graphics throughout the text of your story or analysis; assignments in statistical computing courses involved the creation of a program or script to accomplish some set of tasks.

To be fair, less than a year ago my own assignments for Cal Poly’s course in statistical computing with SAS involved such scripts which were then submitted to be run and checked by me or a grader. The work is tedious and inorganic. Students answer questions as comments in their code; or create a separate document for their answers within which they must now paste code or exported graphics. A non-trivial amount of time gets spent on the report construction, when in fact it should be spent on the programming or writing/interpretation or something more meaningful. Additionally, my assessment of their assignments becomes overly focused on the technical aspects when all they submit is their SAS script file.

To better respect both the students’ time as well as my own, there needs to be a **single** vehicle for the code, output, graphics, and text that go into a report or analysis. With such a tool students would get the training and skills in statistical computing and communication that they deserve and need for today’s employers. Jupyter provides the answer to this dilemma!

Our STAT 330 (Statistical Computing with SAS) course introduces Statistics majors and minors to working in SAS and demands much from them in the way of data management, data manipulation, data visualization and statistical analysis. For most STAT 330 students this is their first programming course, but it is a statistical computing course. Since the focus remains data and statistics, interpretation and report writing are critical components of the course.

Indeed, for students and others using SAS to perform statistical analyses or complete data science projects, the construction of the report remains a key component. I will no longer ask students to submit .sas files for course assignments. The ease with which students can construct dynamic documents, with embedded code and text, with Jupyter notebooks makes other assignment submission types obsolete. Code cells within a Jupyter Notebook should still be commented and documented, but the narrative of the report can exist in the distinct markdown cells.

What's more, planning the arrangement and flow of the document within a Jupyter Notebook need not happen on the fly. Students can experiment and code to their hearts content, annotating and documenting in whatever order and wherever they choose. Cells within the Jupyter Notebook can be re-arranged at any point in time!

Making use of the Jupyter Notebook to write up statistical analyses with embedded SAS code will drastically improve the amount of time students spend communicating about data and statistics, the amount of feedback they get about said communication, and hence the overall quality of their soft skills.

Besides being a wonderful submission vehicle for students, Jupyter Notebooks also provide a nice setting for lecture materials and textbooks themselves; and this is already happening!

CONCLUSION

While traditional report writing using exported graphics and output has gotten easier in recent years, it has been surpassed by Jupyter. Though not standard yet, Jupyter is used by numerous companies and organizations as a key tool for statistics and data science projects. This heavy use along with the ever-increasing demand of employers for people with both technical skills in statistics and programming, and soft skills in communication, necessitates the use of Jupyter as an educational resource and industry tool.

Students' and data scientists' abilities to write clear reports and translate statistical analyses succinctly using numerical and graphical summaries need not be hamstrung by tedious document creation tasks anymore. These abilities are bound to improve with consistent use of the Jupyter Notebook for course assignments, academic publications, data journalism, data science projects in industry and everything in between to create living, dynamic documents!

All of the materials for this Hands-On Workshop, talk, and paper can be found here:

https://github.com/hglanz/SGF2020_JupyterHOW_Glanz

REFERENCES

1. Fuentes, Montse. July 1, 2016. "Reproducible Research in JASA." *AMSTAT NEWS*.
2. "Jupyter." July 21, 2016. Available at <http://jupyter.org>.
3. Granger, Brian. "Jupyter Talks". March 1, 2017. Available at <<https://github.com/ellisonbg/jupyter-talks>>

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Hunter Glanz

Enterprise: Statistics Department, California Polytechnic State University

Address: 1 Grand Avenue

City, State ZIP: San Luis Obispo, CA 93407

Work Phone: 805-756-2792

E-mail: hglanz@calpoly.edu

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.