PharmaSUG 2018 - Paper BB-07

Set-up and usage ways of Jupyter Notebook for SAS®

Alona Bulana, DOCS Global

ABSTRACT

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. It has support for over 40 programming languages, including SAS®. The interactivity of the notebook implemented via so-called kernels. SAS Software released the SAS kernel for Jupyter in September 2016 which is open source. The Notebook itself looks like a chain of cells which may contain either code or Markdown. Code cells can be evaluated and SAS Output or Log will be stored immediately after cell as an inline result. Cells with Markdown are desired for explanations or any other kind documentation. You may ask what are advantages of using a Jupyter Notebook instead of just a SAS code with comments? The answer is that the notebook stores computed results in it which can be recomputed if needed and makes the results completely reproducible. This paper will cover the set-up of Jupyter notebook and examples of its usage: for storing your personal code snippets and for presentations, trainings, or workshops.

INTRODUCTION

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more:

- **Language of choice** The Notebook has support for over 40 programming languages, including Python, R, and SAS.
- **Share notebooks** Notebooks can be shared with others using email, Dropbox, GitHub and the Jupyter Notebook Viewer.
- **Interactive output** Your code can produce rich, interactive output: HTML, images, videos, LaTeX, and custom MIME types.
- **SAS kernel for Jupyter Notebooks** After installing the SAS kernel, you can use a notebook and a SAS installation to write, document, and submit SAS programming statements. The SAS kernel enables Jupyter Notebook to provide the following programming experience:
  - syntax highlighting for SAS programming statements
  - store the input and output from an interactive SAS session

The SAS Kernel for Jupyter Notebook has released in September 2016 and since then began to gain popularity as a document format which is very convenient to use for presentations, training and especially for workshops. Notebooks can be viewed on a computer where SAS is not installed with all produced results but you won't be able to reproduce them — for this SAS is required.

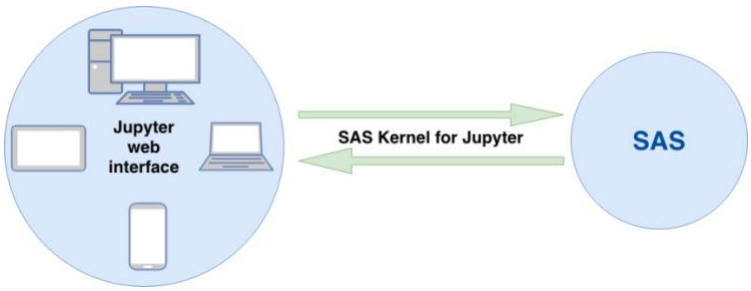
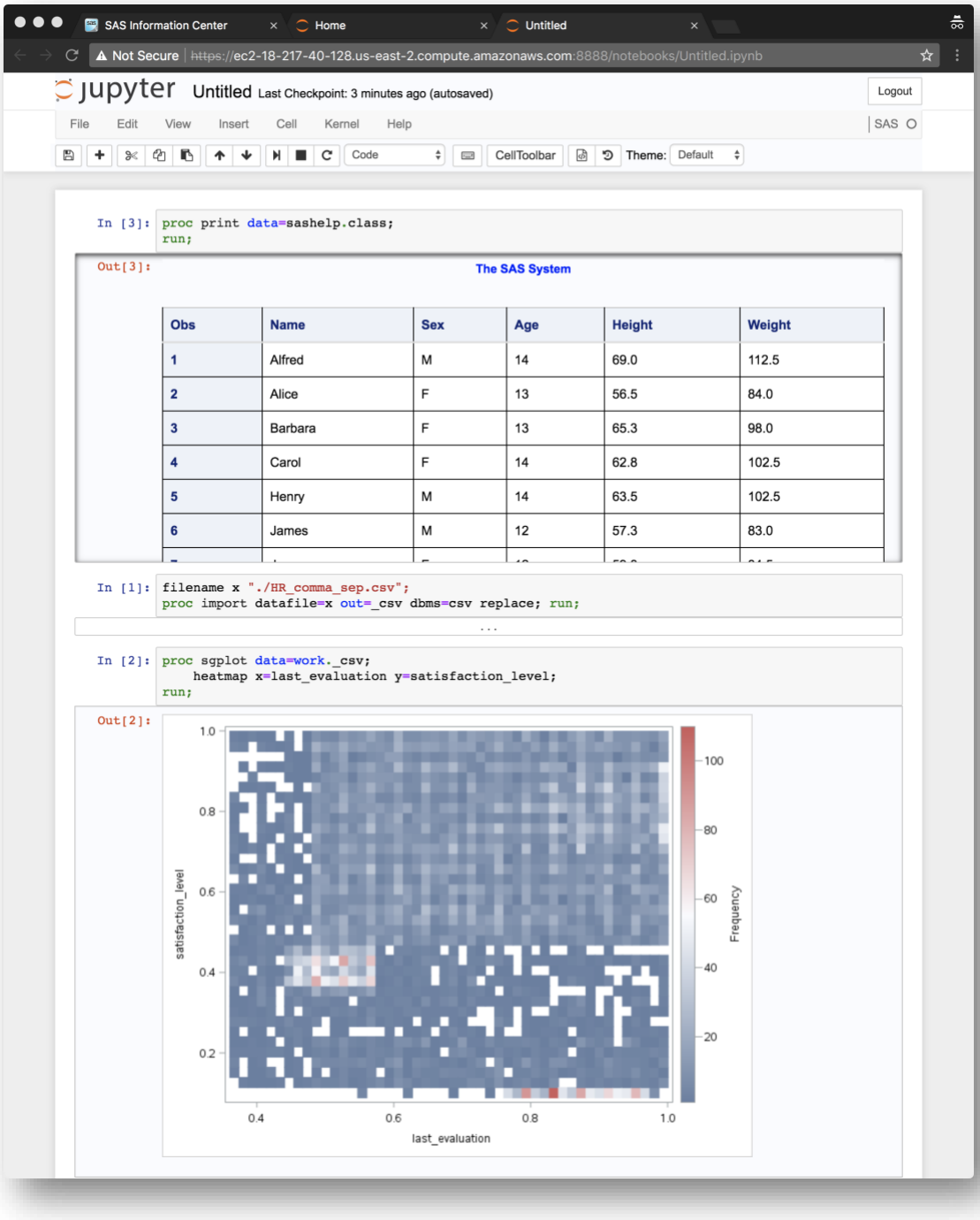


Figure 1: Jupyter architecture



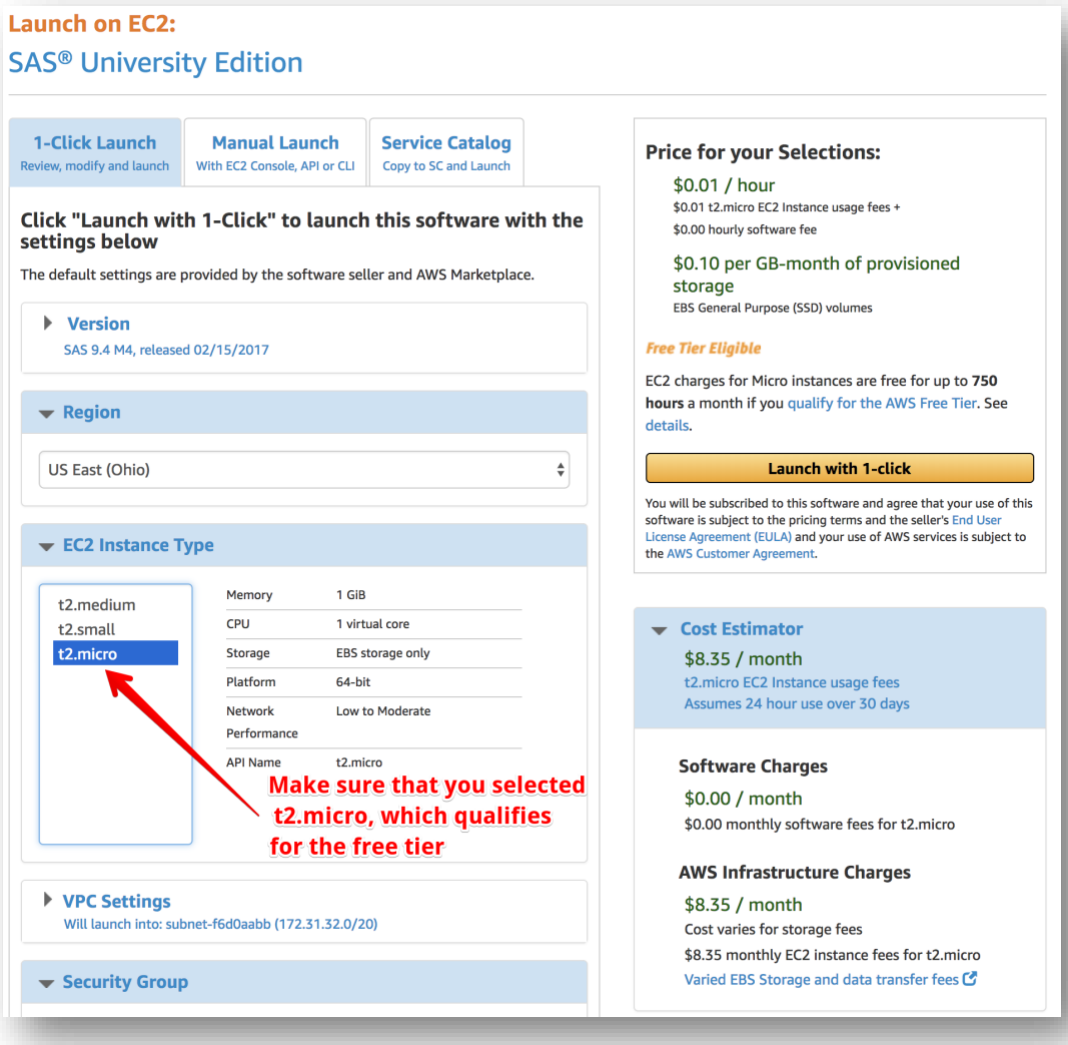
Display 1: The look of the Jupyter Notebook for SAS

This paper will completely cover the installation and set-up of SAS University Edition on the Amazon Web Services — the easiest way to start using Jupyter Notebook for SAS. This is free in certain circumstances — SAS University Edition has no software cost and is eligible for the AWS free tier.

INSTALLING SAS UNIVERSITY EDITION ON THE AMAZON WEB SERVICES

Let's start with installing Installing SAS University Edition on the Amazon Web Services. This free product bundles access to both the SAS Studio and Jupyter Notebook web application.

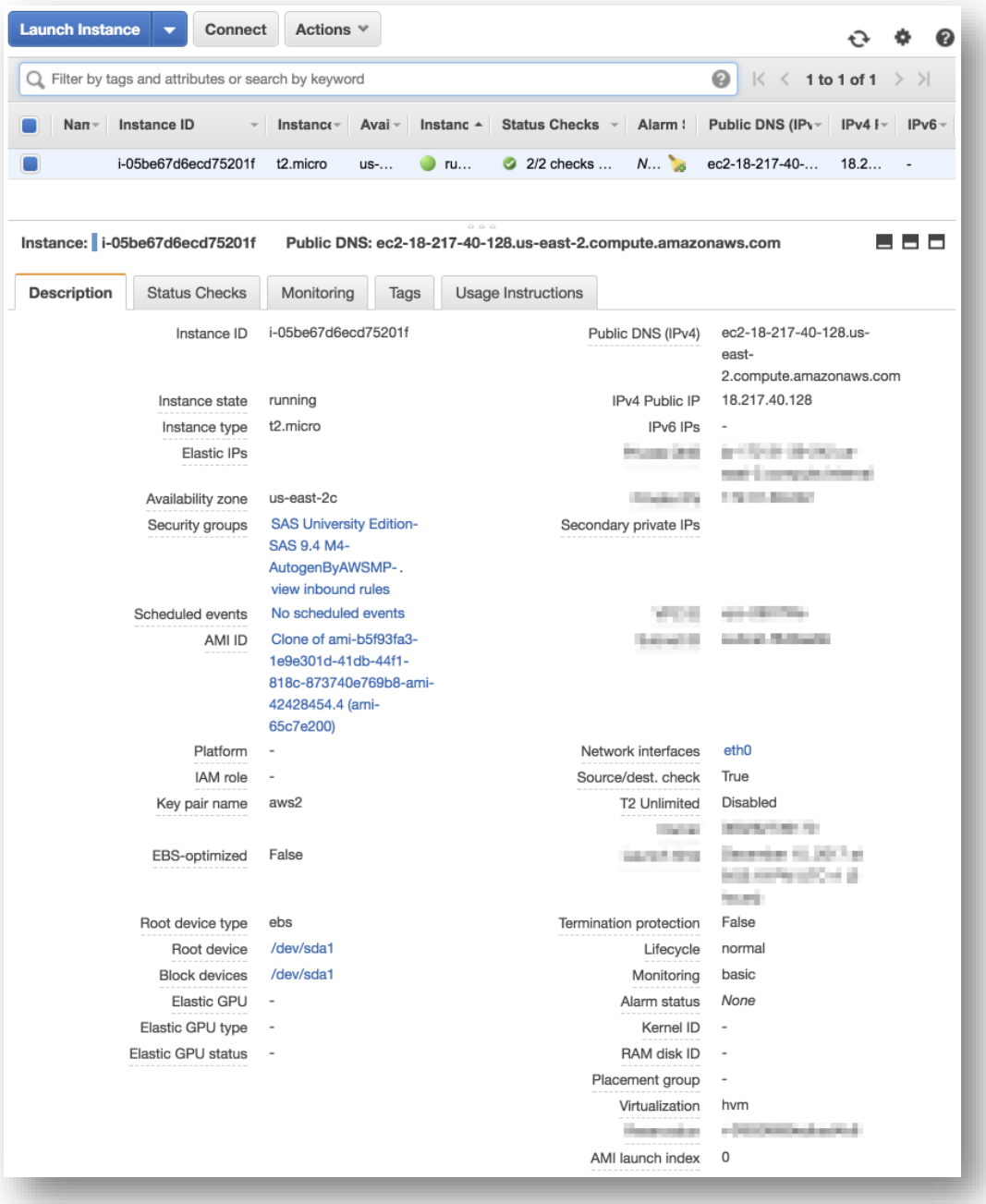
1. **Open the SAS University Edition page.** In the AWS Marketplace, open the page for SAS University Edition. This page displays the instances that are available for the SAS University Edition. You can accept all the default values. Click **Continue**. The Launch on EC2: SAS University Edition page opens.
2. **Sign in to your AWS Marketplace account, if you haven't already.** To sign in or create an account, open the AWS Marketplace. If you are new to AWS, see "Getting Started with AWS". Note: When creating an account, you must provide credit card information to cover any AWS charges. SAS University Edition has no software cost and is eligible for the AWS free tier.
3. **Add SAS University Edition to your list of software subscriptions.** The Launch on EC2: SAS University Edition page lists all of the available instance types for the SAS University Edition. On the **1-Click Launch** tab:
    1. Accept the default value of **t2.micro**, which qualifies for the free tier. If you select a different instance, you might incur AWS charges:



Display 2: Select t2.micro instance

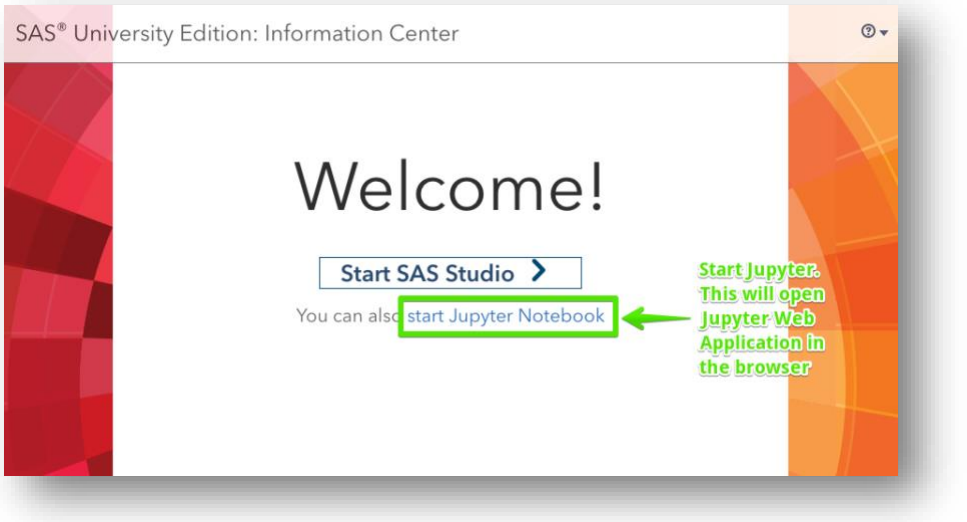
2. Create a key pair, if you have not already.
3. Click Launch with 1-click. A confirmation page notifies you that an instance of SAS University Edition is being launched.

4. **Start SAS University Edition.**
    1. Under the **Related Links** heading on the confirmation page, click **AWS Management Console**. The AWS Management Console opens and lists any instances of SAS University Edition that are associated with your account.
    2. From the table, select a running instance of SAS University Edition. If the value of the Status Checks column is initializing, wait until this initialization is complete before continuing:



Display 3: Instance of SAS University Edition

3. From the **Description** tab, copy the public DNS and paste it in the address bar of a new browser window. In my case it looks like `ec2-18-217-40-128.us-east-2.compute.amazonaws.com`.
4. When prompted, enter `sasdemo` as the user name. For the password, copy and paste the instance ID from the **Description** tab. In my case it looks like `i-05be67d6ecd75201f`. By the way I already destroyed this AWS instance, so the above address/login/password are not valid anymore.
5. The start page of the SAS University Edition has some information on it and two main links:
    1. Start SAS Studio
    2. Start Jupyter Notebook:



Display 4: SAS University Edition start page

5. **Stop or terminate the SAS University Edition instance.**
    1. Open the EC2 Dashboard page.
    2. Under the **Resources** heading, click **Running Instances**.
    3. From the console, select the instance that you want to stop or terminate, and then select **Actions > Instance State**.
        1. Select **Stop** to save the current instance and stop it from running. When prompted, click **Yes, Stop**.
        2. Select **Terminate** to delete the instance. No uploaded data persists. When prompted, click **Yes, Terminate**.
6. **Restart SAS University Edition**. To restart a stopped instance of the SAS University Edition, select **Actions > Instance State > Start** from the AWS Management Console. To restart SAS University Edition if you terminated an instance, you must create a new instance on the SAS University Edition page.

The set-up process is pretty straightforward and in the end, you have a Jupyter accessible via web browser. This is actually good because wherever you are, you can open SAS Studio or Jupyter Notebook — for this you would just need to have a browser.

Another way is to install Jupyter and SAS Kernel manually and then configure it to use your SAS installation. This is explained in details in the Jupyter and SAS Kernel documentation:

- Dependencies
- Installing Jupyter
- Installing the SAS kernel

BRIEF INTRODUCTION TO MARKDOWN IN CONTEXT OF JUPYTER NOTEBOOK

Markdown is a lightweight markup language which is intended to be as easy-to-read and easy-to-write as is feasible. It is important to understand here that any Markdown file is just a plain text file usually with `.md` extension, nothing more. Initially, Markdown's syntax was intended for one purpose: to be used as a format for writing for the web. But it gained a big popularity because of its readability and simplicity and now people use it literally for everything where it is possible. There are a high variety of tools which can convert Markdown to HTML, PDF, TeX, Word documents, PowerPoint slides, etc. One of these tools is pandoc. Many websites have a built-in preview for Markdown files, e.g. https://github.com.

This paper is itself written in Markdown, its source can be found in the GitHub repository for this paper.

Here I will provide some Markdown syntax which will be used later in this paper.

Headers

Headers are set using a hash before the title. The number of hashes before the title text will determine the depth of the header. Header depths are from 1-6:

- H1: `# Header 1`
- H2: `## Header 2`
- H3: `### Header 3`
- H4: `#### Header 4`
- H5: `##### Header 5`
- H6: `###### Header 6`

Text Styling

- Links: **`[Title](URL)`**
- **Bold**: **`**Bold**`** or **`__Bold__`**
- *Italicize*: **`*Italics*`** or **`_Italics_`**
- Lists: **`*`**, **`-`** or **`+`** for every new list item:

```
- item 1
- item 2
    * item 2.1
        + item 2.1.1
...
- item n
```

- Numbered Lists: **`0.`** or **`1.`** for every new list item. Numbering is performed automatically:

```
0. numbered item 1          => 1.
0. numbered item 2          => 2.
    1. numbered item 2.1    => 2.1.
    2. numbered item 2.2    => 2.2.
0. numbered item 3          => 3.
    3. numbered item 3.1 (explicit numbering will be anyway ignored)
    5. numbered item 3.2    => 3.2.
...
```

- Quotes: **`> Quote`**
- Inline Code: **`` `%let str = yellow, 88, green, 0;` ``**

- code blocks:

```
```
/* code blocks goes here */
data _null_;
    array x[*] x1-x12;
    do i = 1 to dim(x);
        x[i] = ranuni(-1);
    end;
run;
```
```

- Images: **`![Image caption text](PATH)`**

At this point, you can see that Markdown cells in a Jupyter Notebook allow users to produce pretty rich formatting with very simple notations. Also, you can refer to local files in your Markdown cell:

```
[subdirectory/]<filename>
```

With this, it is possible to embed images, gifs, and even videos!

For more information, see Daring Fireball's Markdown Syntax and Jupyter Notebook documentation for Markdown Cells.

Navigation between headers

Jupyter adds reference links for each header. This means that you can reference headers using links syntax. This is particularly useful when you have many sections, a Table of Contents and want to build the following navigation features in your notebook:

1. Items in ToC are links to the corresponding sections.
2. Each section has a link **back to top** which will bring a reader to the ToC.

The above concept can be implemented using mentioned above reference links for headers which Jupyter provides out of the box:

```
# Table of Contents

1. [Section 1](#Section-1)              <= link to Section 1
1. [Section 2](#Section-2)              <= link to Section 2
    1. [Section 2.1](#Section-2.1)      <= link to Section 2.1
...
1. [Section n](#Section-n)              <= link to Section n

...

## Section 2.1

**[back to top](#Table-of-Contents)**     <= this will navigate you back
                                             to the Table of Contents

...
```

In the following sections, we will consider two real-world examples of using Jupyter Notebook.

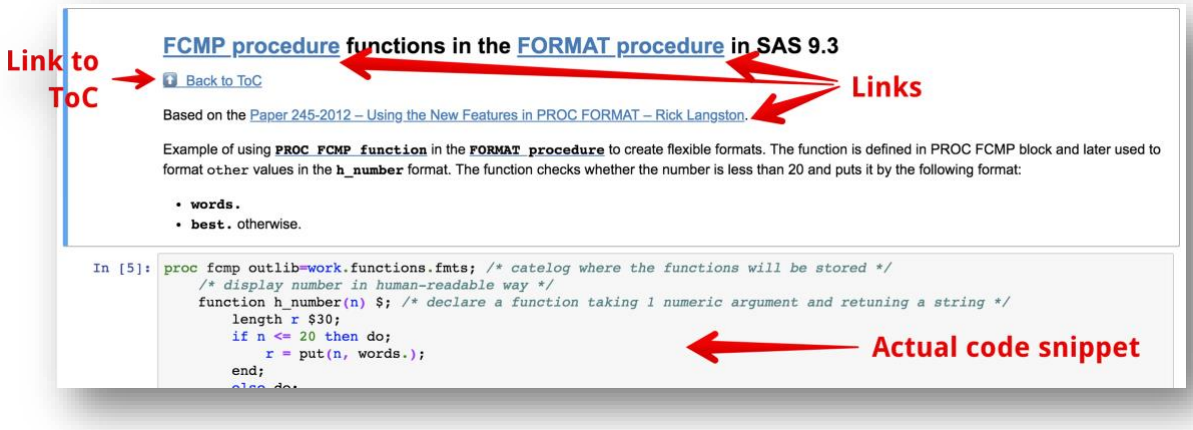
USING JUPYTER FOR STORING YOUR CODE SNIPPETS

SAS language has a very rich syntax, a lot of procedures, and many options. Sometimes it certainly is hard to recall a syntax of something you programmed before. And here code snippets come to the rescue. Code snippets are small blocks of reusable code that are used among many programs. I suppose almost all of us have a collection of code snippets that we use in our everyday programming activities. Many people may remind me that we've got code snippets in SAS Studio, but here I'm talking about those code snippets which you usually expect to see together with results for a better overview. In certain situations, code snippets need some explanation and details which usually are incorporated into comments.

Jupyter Notebook is a great tool for storing and maintaining your SAS code snippets. Cell structure of the notebook allows us to semantically separate snippets one from each other. A snippet can be executed immediately and results will be stored in the notebook, so that is indeed helpful when you see its SAS Output or Log together with the code snippet.

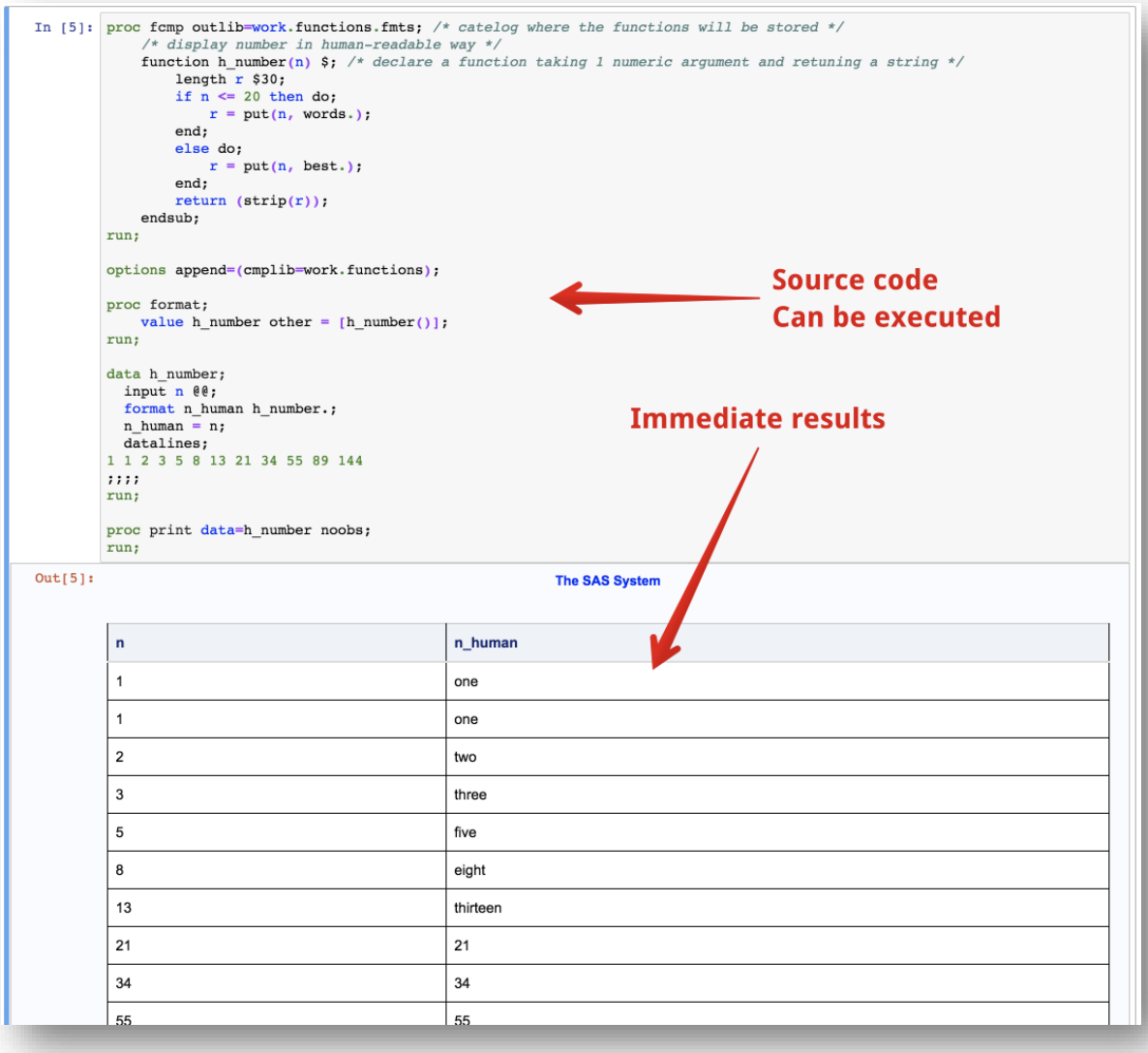
For demonstration purposes, I created an example notebook with several code snippets. Its source code can be found in the GitHub repository for the paper. GitHub provides a preview for Jupyter Notebooks, but you cannot execute anything there. Header links also do not work on GitHub. To make all these things work you need to open this notebook in Jupyter.

1. Making explanation in Markdown gives you possibility to compose informative text in fact. You can support your explanation by links, lists or highlight some important parts using bold:



Display 5: Markdown notes in Jupyter Notebook

2. Code and results. Results can be either SAS Output or Log if no output was produced:



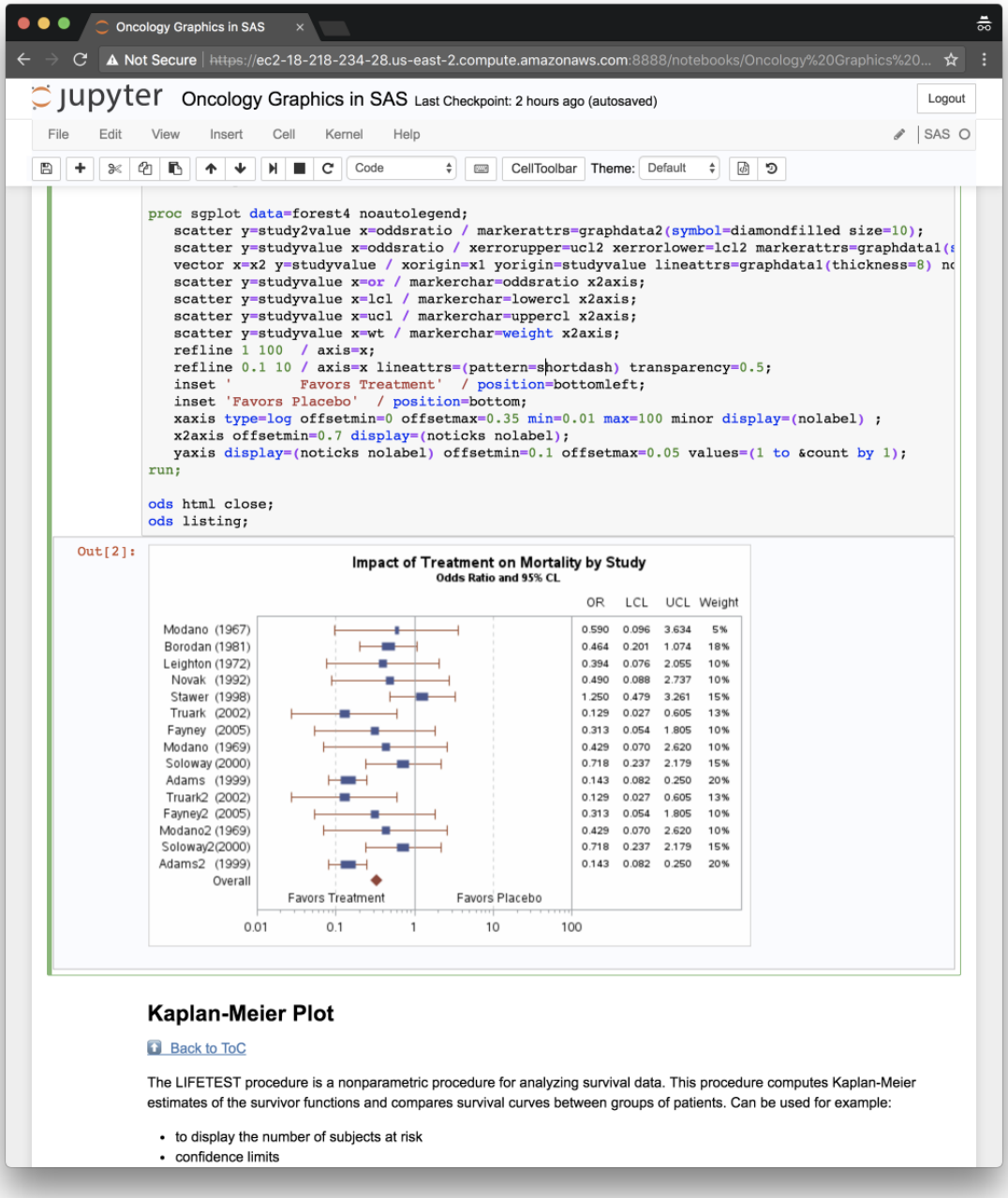
Display 6: Viewing inline results

I hope this gave you an idea for how you can begin using Jupiter notebooks, they're really becoming popular and it's a great way to explore your data and your code an interactive way, it's great for displaying plots and charts and all kind of different things. So it's definitely something that's nice to know how to use.

DOING A WORKSHOP IN JUPYTER

Jupyter is a great tool for conducting workshops and trainings. The usual workshop or training structure is a chain of explanations and live code walkthroughs. And Jupyter fits ideal for this task.

1. Step by step code, notes, text, equations or media content are organized using cells.
2. Markdown provides a rich text formatting for better explanation experience. Links are particularly useful when you can immediately follow them add view any supplemental content.
3. Notebooks can be shared – so users may follow the instructor at the same time or repeat the results later. Therefore, an audience can focus on the content instead of the maintenance of their notes.



Display 7: Oncology Graphics Workshop example

Jupyter Notebook for the above example can be found in the GitHub repository for the paper.

This is extremely useful, so this is where we really start to see why these notebooks are becoming really popular, because to be able to show plots and things like that as we're stepping through our code — that definitely helps you to explore your data and a code in real time without needing to return to your SAS programming environment.

CONCLUSION

Jupyter Notebook is a great open-source tool for creating and sharing documents that combine live code with narrative text, mathematical equations, visualizations, interactive controls, and other rich output. It has many successors and extensions like JupyterLab and JupyterHub which are actively developed and used among data science community. If you do any workshops, trainings or documentation you should try using Jupyter Notebook with SAS and I promise to you that you will love it.

REFERENCES

Project Jupyter website. (Accessed March 2018) Available at http://jupyter.org/index.html

Jupyter Notebook documentation. (Accessed March 2018) Available at https://jupyter-notebook.readthedocs.io/en/stable/index.html

sassoftware/sas_kernel. GitHub repository. *SAS Kernel for Jupyter*. (Accessed March 2018) Available at https://github.com/sassoftware/sas_kernel

SAS Kernel for Jupyter documentation. (Accessed March 2018) Available at https://sassoftware.github.io/sas_kernel/

sassoftware/saspy. GitHub repository. *A Python interface module to the SAS System*. (Accessed March 2018) Available at https://github.com/sassoftware/saspy

SAS Institute Inc. 2016. *SAS® University Edition: Quick Start Guide for Amazon Web Services Marketplace*. Cary, NC: SAS Institute Inc. (Accessed March 2018) Available at http://support.sas.com/software/products/university-edition/docs/en/SASUniversityEditionQuickStartAWS.pdf

Amazon Web Services, Inc. 2018. *Getting Started with AWS*. (Accessed March 2018) Available at https://aws.amazon.com/getting-started/?nc2=h_l2_cc

ACKNOWLEDGMENTS

I want to thank Project Jupyter team for their great work on Jupyter.

CONTACT INFORMATION

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

Alona Bulana
DOCS Global
Berlin, Germany
alonabulana at gmail dot com (email)