Paper 3133-2019

Learning Data Science with SAS® University Edition and JupyterLab

Brian R. Gaines, SAS Institute Inc., Cary, NC

Abstract

One of the interfaces that SAS® University Edition includes is the popular JupyterLab interface. You can use this open-source interface to generate dynamic notebooks that easily incorporate SAS® code and results into documents such as course materials and analytical reports. The ability to seamlessly interweave code, results, narrative text, and mathematical formulas all into one document provides students with practical experience in creating reports and effectively communicating results. In addition, the use of an executable document facilitates collaboration and promotes reproducible research and analyses. After a brief overview of SAS University Edition, this paper describes JupyterLab, discusses examples of using it to learn data science with SAS, and provides tips. SAS University Edition, which is available at no charge to educators and learners for academic, noncommercial use, includes SAS® Studio, Base SAS®, SAS/STAT®, and SAS/IML® software and some other analytical capabilities.

Introduction

In today's increasingly connected world, massive amounts of data are being continuously collected from a multitude of devices and sensors. As more and more businesses and organizations seek to derive useful insights from these data, there has been a surge in demand for people who have data science and analytics skill sets. Although "data scientist" (Glassdoor 2019) and "statistician" (U.S. News and World Report 2019) routinely rank as two of the top jobs in the United States, employers have struggled to find enough candidates with the desired skills (Deloitte 2016). In response, new academic programs in analytics and data science have sprung up. There have also been calls for existing academic programs in statistics to be updated to reflect the changing technological landscape. These recommendations include an increased emphasis on computing, communication skills, and reproducible analyses (De Veaux et al. 2017; GAISE College Report ASA Revision Committee 2016; Horton and Hardin 2015, and references therein). This paper demonstrates how these points of emphasis can be readily addressed with SAS University Edition and JupyterLab.

The remainder of the paper is organized as follows. The section "Free Resources for Educators and Learners" provides an overview of the various free resources available from SAS for educators and learners. These resources include lecture materials, e-learning courses, and software. The section "Jupyter Notebooks" presents background information about Jupyter notebooks and describes the basics of using them in JupyterLab with SAS University Edition. The section "JupyterLab" highlights some of the new features available in JupyterLab and compares them to those of its predecessor, Jupyter Notebook. The section "Example Jupyter Notebooks" presents three examples that demonstrate how you can use JupyterLab and SAS University Edition to effectively teach and learn data science. The examples include a homework assignment, lecture notes, and an analytical report that were created in JupyterLab. Finally, the section "Tips for Using JupyterLab" offers a few tips to help you use JupyterLab with SAS more efficiently.

Free Resources for Educators and Learners

To help address the well-documented analytics skills gap, SAS offers an abundance of resources for educators and learners that you can access from the SAS Academic Programs website at `http://www.sas.com/academic`, including the following:

- free lecture materials that you can download and incorporate into your courses, in addition to curriculum consulting
- four free e-learning courses:
  - "SAS Programming 1" covers the basics of using SAS and is a prerequisite for other e-learning courses.
  - "Statistics 1" covers a variety of statistical techniques, including analysis of variance (ANOVA), linear regression, and logistic regression, and how to implement them with SAS.

  - "SAS Programming for R Users" introduces SAS to experienced R programmers.
  - "SAS Viya Enablement" provides an overview of SAS® Viya®, which is a new, cloud-enabled, in-memory analytics engine from SAS that uses distributed computing for highly scalable, very fast execution.
- a library of free video tutorials
- access to free SAS software through either SAS® OnDemand for Academics or SAS University Edition

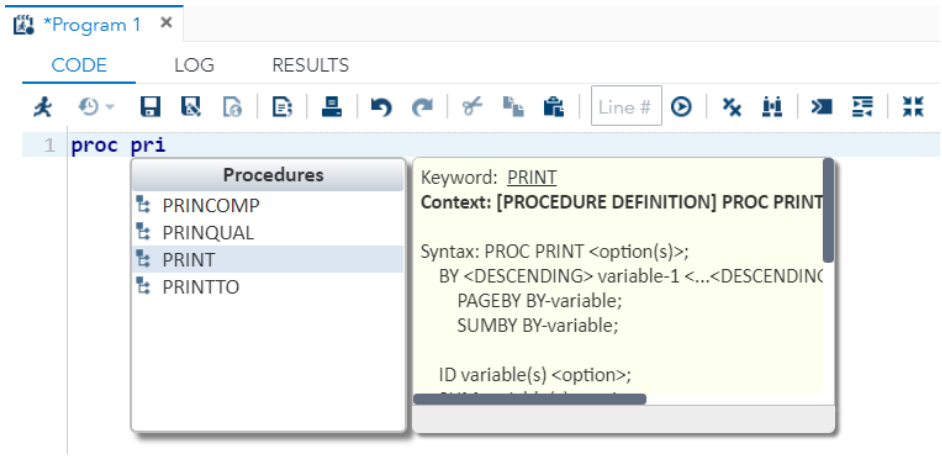
SAS OnDemand for Academics is a cloud-based offering that alleviates the need to install anything locally, but it does require an internet connection. It includes access to products such as SAS® Enterprise Guide®, SAS Studio, SAS® Enterprise Miner™, and SAS® Forecast Server. Professors can create courses that students can enroll in to streamline the sharing of data and code. For tips and additional information about using SAS OnDemand for Academics, see Mullis (2018).

SAS University Edition

SAS University Edition is a free version of select SAS products that runs on all three major operating systems. It can be downloaded from the SAS website and installed on your personal computer, or it can be accessed through Amazon Web Services. Despite its name, SAS University Edition is not just for students and is actually available to anyone who wants to learn and use SAS software for noncommercial purposes. It includes the main products that you need for performing advanced statistical analysis, including Base SAS, SAS/STAT, SAS/IML, and some forecasting procedures.

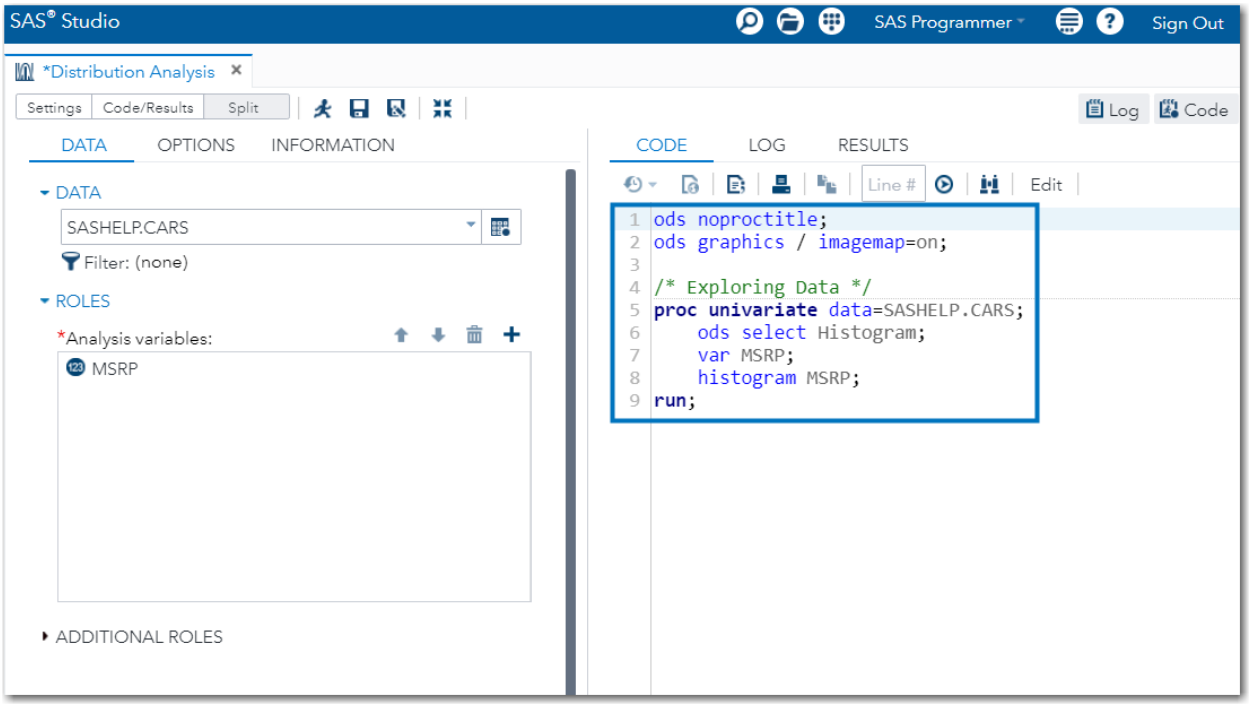
One of the interfaces that SAS University Edition includes is SAS Studio. The code editor in SAS Studio has several tools to increase your productivity when you are developing SAS code. For example, syntax autocomplete and pop-up help are available to help you code faster (Figure 1).

Figure 1 Code Autocomplete and Pop-Up Tips in SAS Studio



In addition to furnishing a code editor, SAS Studio also provides graphical user interfaces to many of the commonly used analytical methods, which are presented as "tasks." These tasks enable new users to obtain results much more quickly and easily. The underlying SAS code is automatically generated in real time and can be used as a learning tool, a building block for further analysis, or a part of a larger analytical pipeline (Figure 2). Tasks can be saved, customized to fit your specific needs, and shared with colleagues. For more information about SAS Studio, see the SAS Studio Customer Product page at `http://support.sas.com/software/products/sas-studio/index.html`. For more information about SAS Studio tasks and developing custom tasks, see SAS Institute Inc. (2018b), Dexter et al. (2016), and Inman and Wright (2017).

Figure 2 SAS Studio Task Real-Time Code Generation



Jupyter Notebooks

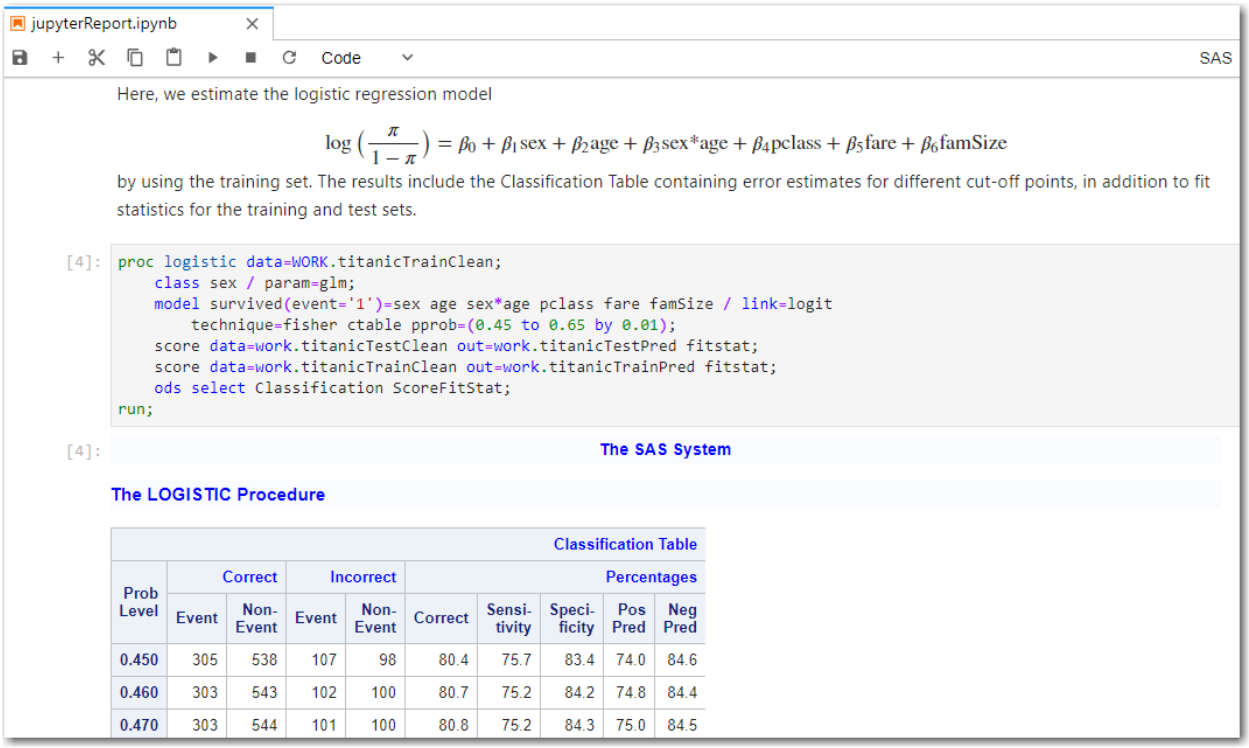
Background

SAS University Edition also comes with a second integrated development environment (IDE), JupyterLab. Before discussing JupyterLab, let's focus on its primary programming interface, Jupyter notebooks. Jupyter notebooks are also the focal point of JupyterLab's predecessor, Jupyter Notebook (notice the capital "N"). Jupyter notebooks are an open-source, browser-based interface that support a wide variety of programming languages via analytical engines called *kernels*. The SAS kernel itself is an open-source project from SAS on GitHub that can also be used with a SAS license (SAS Institute Inc. 2019b). In recent years, Jupyter notebooks have become very popular in the data science community because they enable you to seamlessly interweave code, results, narrative text, and equations into one document (Figure 3).

For example, an important stage in the data analysis life cycle is communicating your results. Logistically, a traditional workflow involves using statistical software to obtain the results, and then copying and pasting these results into another program to create a report or presentation. Baumer et al. (2014) refer to this common workflow as the "copy-and-paste paradigm." Not only is the copy-and-paste workflow tedious, but it is also more prone to errors and ethics violations (Baumer et al. 2014). The workflow of Jupyter notebooks, on the other hand, eliminates the need to copy and paste the results because they appear directly in the notebook alongside explanatory text (Figure 3).

Another benefit of using Jupyter notebooks instead of the copy-and-paste approach is that Jupyter notebook documents are reproducible. In recent years, much attention has been devoted to the apparent prevalence of irreproducibility in scientific research (McNutt 2014; Baker 2016). Although the problem has many causes, a copy-and-paste workflow makes it more difficult to reproduce a data analysis. The ability to include executable code in a Jupyter notebook makes reproducibility much easier. This also facilitates collaboration with others, which is another important skill for a data scientist to develop (De Veaux et al. 2017).

Figure 3 A Jupyter Notebook with Narrative Text, Code, and Results



jupyterReport.ipynb

Code SAS

Here, we estimate the logistic regression model

$$\log\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \beta_1 \text{sex} + \beta_2 \text{age} + \beta_3 \text{sex*age} + \beta_4 \text{pclass} + \beta_5 \text{fare} + \beta_6 \text{famSize}$$

by using the training set. The results include the Classification Table containing error estimates for different cut-off points, in addition to fit statistics for the training and test sets.

[4]:
```
proc logistic data=WORK.titanicTrainClean;
    class sex / param=glm;
    model survived(event='1')=sex age sex*age pclass fare famSize / link=logit
        technique=fisher ctable pprob=(0.45 to 0.65 by 0.01);
    score data=work.titanicTestClean out=work.titanicTestPred fitstat;
    score data=work.titanicTrainClean out=work.titanicTrainPred fitstat;
    ods select Classification ScoreFitStat;
run;
```

[4]: The SAS System

The LOGISTIC Procedure

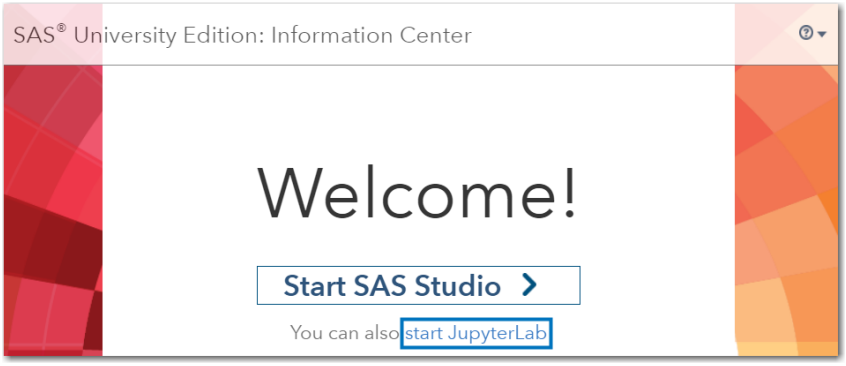
Classification Table

Prob Level	Correct		Incorrect		Percentages				
	Event	Non-Event	Event	Non-Event	Correct	Sensi-tivity	Speci-ficity	Pos Pred	Neg Pred
0.450	305	538	107	98	80.4	75.7	83.4	74.0	84.6
0.460	303	543	102	100	80.7	75.2	84.2	74.8	84.4
0.470	303	544	101	100	80.8	75.2	84.3	75.0	84.5

Basics

Now that you understand some of the benefits of Jupyter notebooks, let's look at some of the basics of using them in JupyterLab within SAS University Edition. After you launch SAS University Edition through a virtual machine and the SAS University Edition: Information Center opens, click **start JupyterLab** under **Start SAS Studio** (Figure 4).

Figure 4 SAS University Edition: Information Center



JupyterLab opens in a new browser tab and initially displays a file browser in the collapsible left sidebar in addition to a **Launcher** tab in the main work area (Figure 5). The **Launcher** tab enables you to open a notebook that uses one of the available kernels. In addition to the SAS kernel, SAS University Edition also includes the Python 3 kernel that contains the SASPy module. The SASPy module is an open-source interface to SAS that enables you to access SAS from Python. For more information about SASPy, see SAS Institute Inc. (2019a) and Hemedinger (2017). To open a notebook that uses the SAS kernel, click the **SAS** button in the Notebook section of the **Launcher** tab.

A notebook consists of different blocks, called *cells*, which by default enable you to run code that uses the selected kernel. For example, if you type SAS code into a code cell and press CTRL+ENTER, the code is executed by SAS and the output is displayed directly below the code chunk (Figure 6).

Figure 5 The JupyterLab Interface

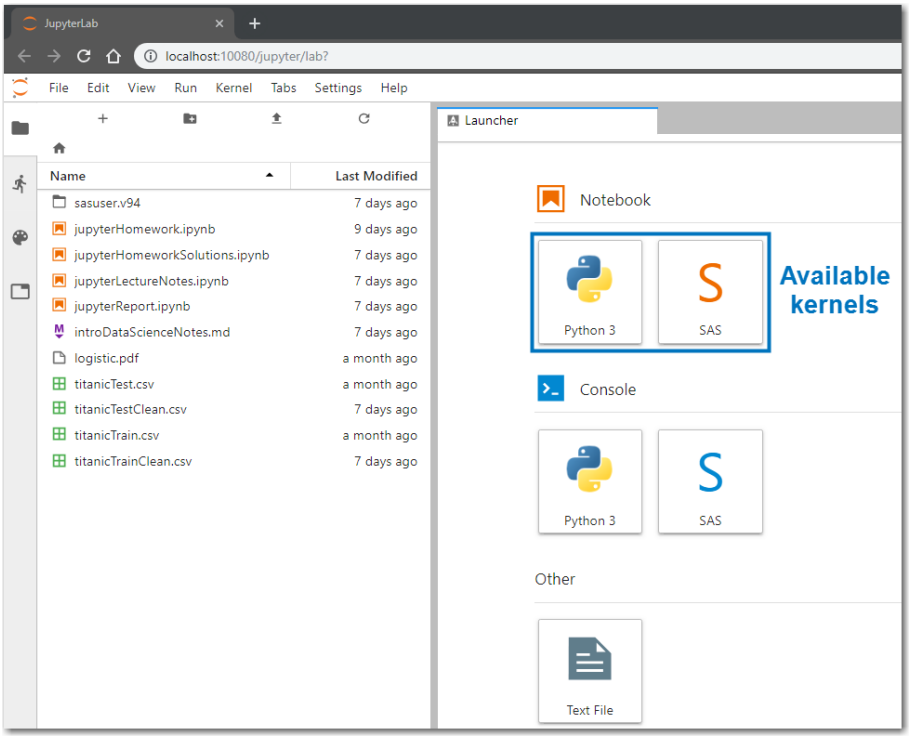
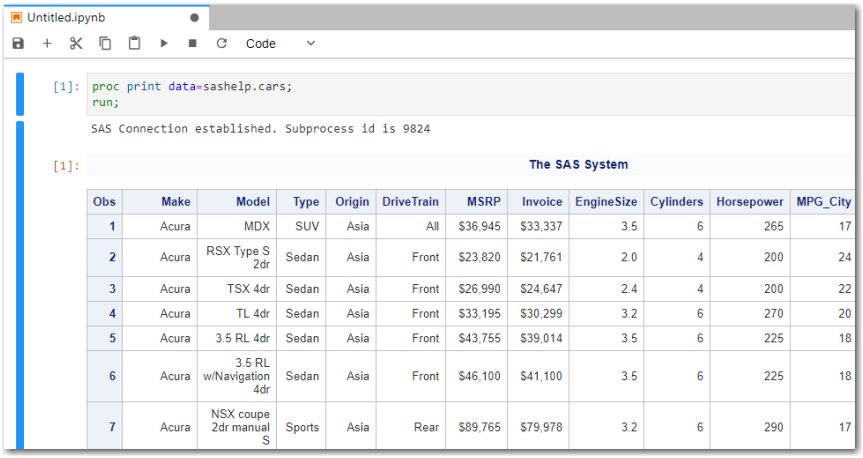


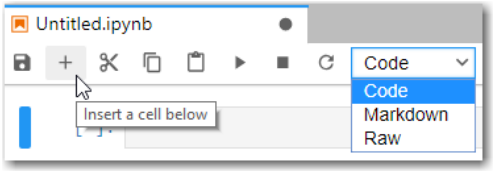
Figure 6 Output Directly Below Executed Code



Obs	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	Cylinders	Horsepower	MPG_City
1	Acura	MDX	SUV	Asia	All	$36,945	$33,337	3.5	6	265	17
2	Acura	RSX Type S 2dr	Sedan	Asia	Front	$23,820	$21,761	2.0	4	200	24
3	Acura	TSX 4dr	Sedan	Asia	Front	$26,990	$24,647	2.4	4	200	22
4	Acura	TL 4dr	Sedan	Asia	Front	$33,195	$30,299	3.2	6	270	20
5	Acura	3.5 RL 4dr	Sedan	Asia	Front	$43,755	$39,014	3.5	6	225	18
6	Acura	3.5 RL w/Navigation 4dr	Sedan	Asia	Front	$46,100	$41,100	3.5	6	225	18
7	Acura	NSX coupe 2dr manual S	Sports	Asia	Rear	$89,765	$79,978	3.2	6	290	17

You insert additional cells into the notebook by using the plus-sign button on the notebook's toolbar. The toolbar also has a drop-down menu that lets you change the cell type (Figure 7).

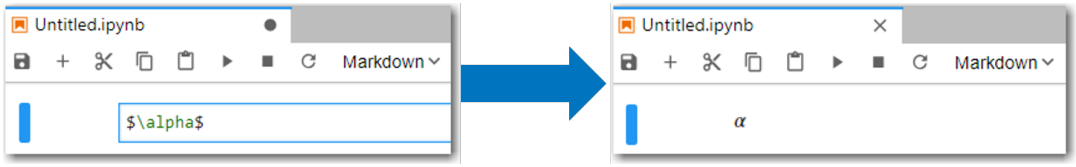
Figure 7 Jupyter Notebook Toolbar



Markdown cells are the other primary cells in a notebook. Markdown is a simple but effective syntax that can be used for explanatory text, section headings, and mathematical notation. You can include math symbols by using LaTeX math syntax. For example, α produces α (Figure 8). See Jupyter Team (2015) for more information about Markdown syntax. You can slowly combine code cells with their corresponding output and Markdown cells with formatted text to create your analytical notebook. The section "Example Jupyter Notebooks" discusses a few use

cases and examples. After you finish constructing your notebook, you can use the **File** menu to download it as a notebook file (file extension .ipynb) to share as a living, executable document, or you can export it and share it as another file type, such as HTML.

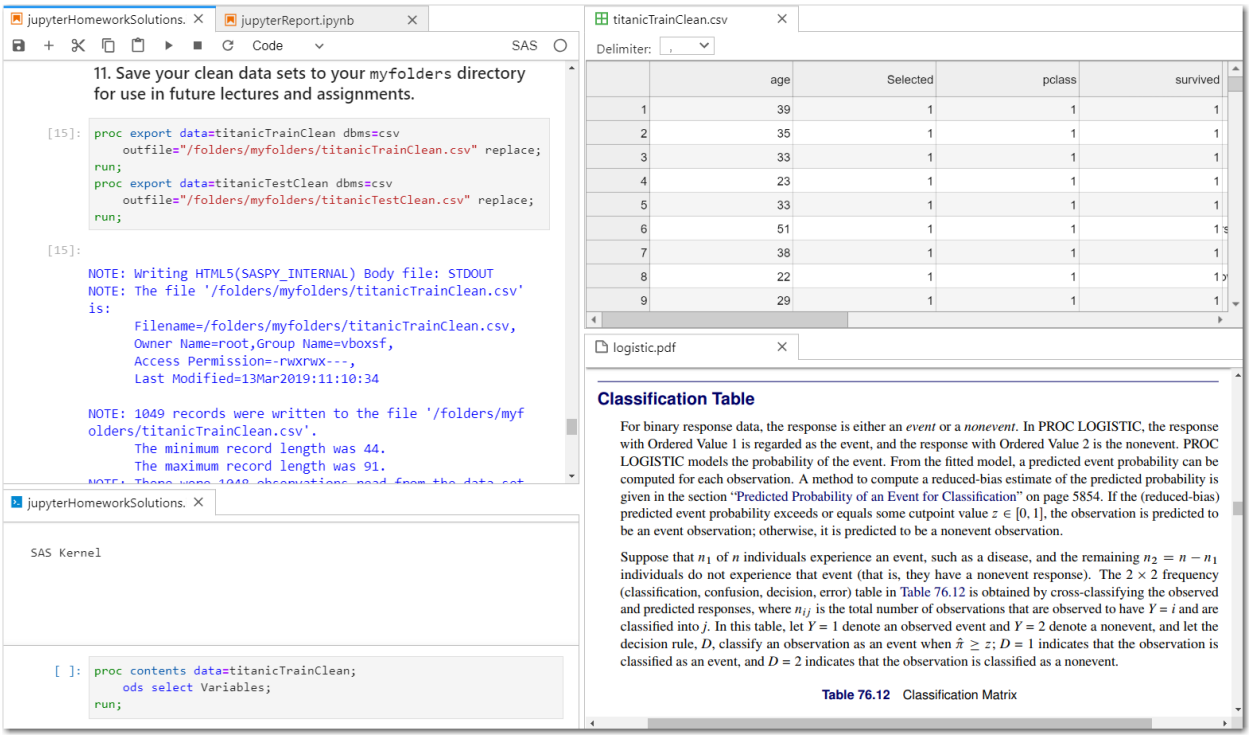
Figure 8 Mathematical Notation in Markdown Cells



JupyterLab

JupyterLab is the successor to the Jupyter Notebook application, and it contains additional features and flexibility. JupyterLab conveniently arranges several elements from Jupyter Notebook, such as the file browser, notebooks, and a text editor, into one interface. Code consoles that enable you to interactively run code for a kernel have been added, in addition to support for viewing other types of files, such as PDF and CSV files. Multiple notebooks, code consoles, and other supported files now occupy different tabs within the main work area of JupyterLab, as opposed to being spread across multiple windows or browser tabs. You can flexibly resize and rearrange the tabs in the work area (Figure 9).

Figure 9 Flexible Main Work Area in JupyterLab



The notebook interface within JupyterLab has also been enhanced, but do not worry: it is compatible with the notebooks that you created in Jupyter Notebook. The notebook editor includes the following enhancements:

- Cells are collapsible to help navigate long notebooks and code that generates lengthy output.
- You can drag and drop multiple cells to rearrange the notebook's layout.
- You can create more than one view of the same notebook so that you can easily work on different parts of a notebook without scrolling back and forth.
- You can link a code console to a notebook, and the code console has the same workspace as the notebook.

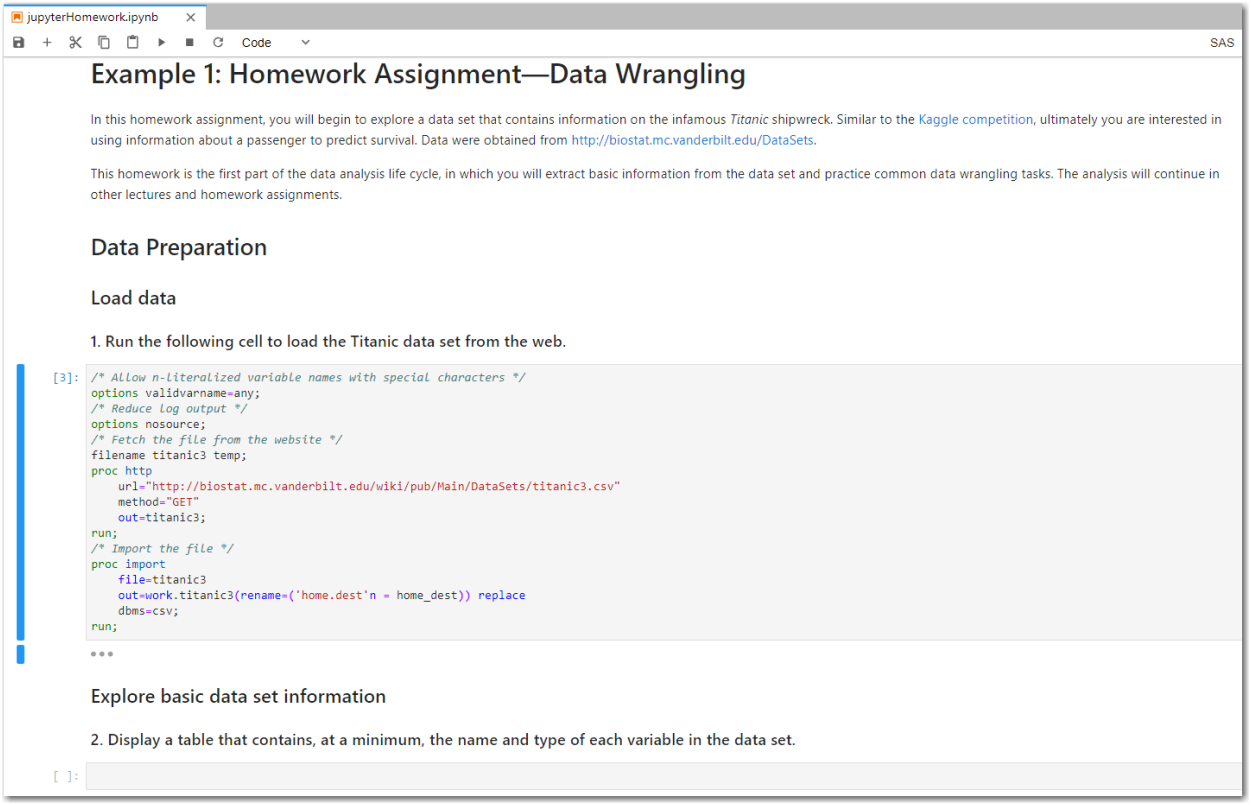
Example Jupyter Notebooks

This section presents three different examples that highlight the power of using JupyterLab with SAS University Edition to teach and learn data science. You can download the notebook files from the SAS® Global Forum 2019 GitHub repository, located at `http://github.com/sascommunities/sas-global-forum-2019`.

Example 1: Homework Assignment—Data Wrangling

The first example of how you can use JupyterLab to learn and teach data science with SAS is a homework assignment on data management and preprocessing. By some accounts, as much as 80% of a data scientist's time is devoted to managing and cleaning data (Press 2016). Therefore, it is imperative for an aspiring data scientist or statistician to obtain these skills. As a result, experience in managing and cleaning large, messy data sets has been increasingly incorporated into statistics and data science courses to better prepare students for the workforce (Hardin et al. 2015; Horton, Baumer, and Wickham 2015). One skill to this end is knowledge of relational databases and how to query them by using the ubiquitous Structured Query Language (SQL). You can accomplish this in SAS by using the SQL procedure, which is demonstrated in this example (Figure 10).

Figure 10 Homework Assignment in JupyterLab



In the homework assignment, students are first asked to submit a variety of PROC SQL queries to obtain basic information about a data set that contains passenger information from the infamous *Titanic* shipwreck.[1] Students are also required to perform other common data wrangling tasks, such as creating training and tests sets and addressing missing values. The assignment is designed to mimic the initial exploration and cleaning of a data set within the broader data analysis cycle. As students discover in the assignment, the data set that is used in the example contains 1,309 observations and thus is not terribly large. However, the assignment is worthwhile because the process would be the same if you were querying larger data tables that are stored in a relational database management system.

There are many benefits to creating a homework assignment as a Jupyter notebook file. As demonstrated in this example, instructors can provide the scaffolding for the homework notebook and can optionally include time-saving code, which enables students to focus on the main points of the assignment (Figure 10). The scaffolding and starter code can be used as training wheels to reduce the learning curve for Jupyter notebooks and SAS syntax. The amount

[1] The data were obtained from http://biostat.mc.vanderbilt.edu/DataSets.

of starter code can be reduced over time as students become more familiar with these technologies. The instructor can also distribute the solutions to the assignment as a completed version of the notebook. In addition, as discussed earlier, the notebook interface provides a streamlined workflow in which students do not need to copy and paste between SAS Studio and a word processor to complete the homework assignment. Besides being environmentally friendly, submitting homework electronically as a Jupyter notebook is also convenient for the grader, because everything is in one document and the code can be readily executed to help identify potential errors.

Example 2: Interactive Lecture Notes—Bootstrap Confidence Intervals

Another excellent use for JupyterLab in education is to create lecture notes. As in Example 1, using a notebook enables you to abandon the copy-and-paste approach to integrating code and results into a document. However, now the instructors are the primary beneficiaries. Although there is an up-front cost to convert existing lecture notes to notebooks, in the long run this makes it much easier to modify any code or examples in the notes. It also provides students with example notebooks that help them learn the notebook interface, and it enables them to explore the code and examples from class more easily (Hicks and Irizarry 2018). Plus it promotes an ethos of reproducibility that many prominent educators have been advocating for (Baumer et al. 2014; Horton, Baumer, and Wickham 2015; Çetinkaya-Rundel and Rundel 2018).

This example provides a snippet of lecture notes on using the bootstrap to construct confidence intervals (Figure 11). From a practical standpoint, resampling methods, such as randomization tests and the bootstrap, are important tools for a data scientist's toolkit (Hesterberg 2015). There is also evidence that students who are taught using these methods are better able to grasp important statistical concepts (Chance, Mendoza, and Tintle 2018; Tintle et al. 2018). Hesterberg (2015) provides an excellent overview of the bootstrap and highlights several of its educational benefits, including the following:

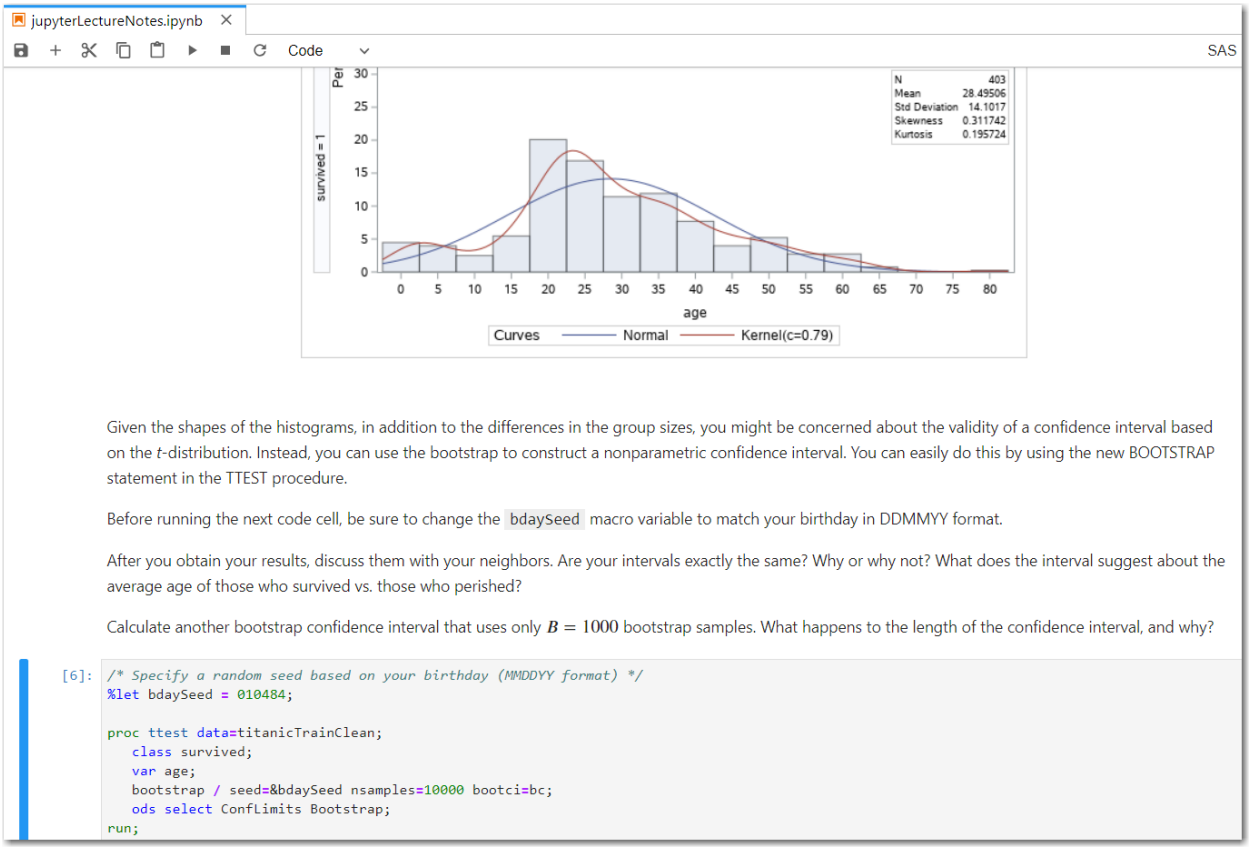
- Important but abstract concepts such as sampling distributions, standard errors, the central limit theorem, and confidence intervals are made more concrete through plots of the bootstrap distribution.
- Students are provided with a general framework that applies to a wide variety of statistics, so they can focus on ideas instead of formulas. They can also work with other, possibly more appropriate statistics rather than only those that are "well behaved."
- The bootstrap can be used to better understand formula methods, and it serves as a method for students to check their work.

There are a variety of ways in which the bootstrap can be performed in SAS; see Wicklin (2018) for an outstanding guide. Of note is the BOOTSTRAP statement that was added to the TTEST procedure in SAS/STAT 14.3 and is available in the latest version of SAS University Edition. In addition to providing bootstrap standard error and bias estimates, the BOOTSTRAP statement enables you to easily construct a variety of bootstrap confidence intervals for one-sample, paired, and two-sample designs (SAS Institute Inc. 2018c).

In this example, which uses the same data set as Example 1 for the sake of continuity, PROC TTEST is used to construct a bootstrap confidence interval for the difference in the means of two groups. The lecture notes are designed to be distributed to students before class so they can follow along on a laptop and submit the code. Specifically, at the appropriate time, each student is asked to construct a bootstrap confidence interval that uses an individual random seed, such as his or her birthday (Figure 11). Students are then asked to briefly discuss their results in small groups before the results are discussed as a class.

Not only does this brief activity break up the monotony of a traditional lecture, but it also reiterates important statistical ideas, such as sampling variability and sampling distributions (Hesterberg 2015). Also, as demonstrated by this example, distributing lecture notes as Jupyter notebook files before class facilitates the use of active learning during class (Hicks and Irizarry 2018). In active learning, students engage in activities during class instead of passively listening to a lecture. As highlighted by a recent analysis of more than 200 studies, a growing body of literature suggests that active learning is more effective than traditional lectures, especially in the fields of science, technology, engineering, and mathematics (STEM) (Freeman et al. 2014). Active learning is also one of the six recommendations in the Guidelines for Assessment and Instruction in Statistics Education (GAISE) College Report endorsed by the American Statistical Association (GAISE College Report ASA Revision Committee 2016).

Figure 11 Interactive Lecture Notes in JupyterLab



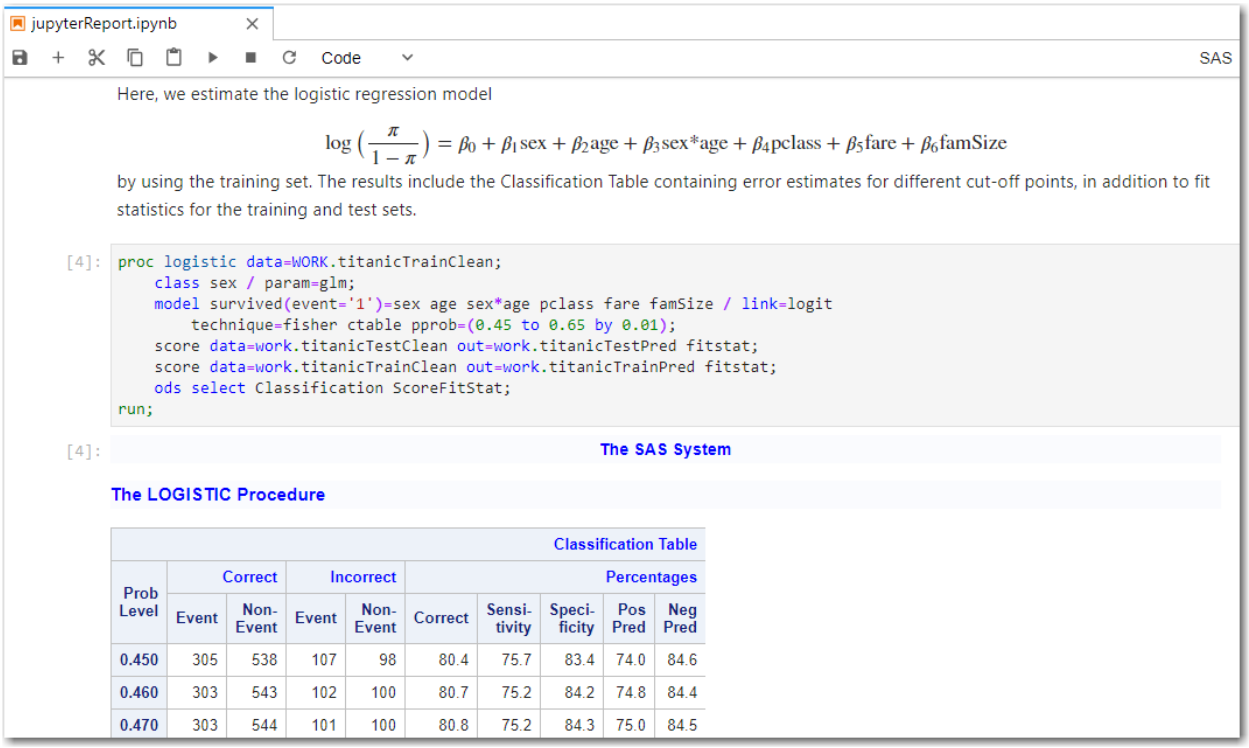
Example 3: Analytical Report—Final Project

Another crucial skill for a statistician or data scientist is the ability to effectively communicate the results of an analysis. This example demonstrates how you can do this efficiently with JupyterLab. The example is an excerpt from a final project, but it is representative of an analytical report for a colleague or client. The hypothetical final project builds on the data wrangling and exploratory analysis that were completed in previous assignments and lectures. This approach exposes students to the entire data analysis life cycle, which is a key part of learning data science (Hardin et al. 2015; De Veaux et al. 2017).

The ultimate goal of the project is to build a classifier to predict the survival of *Titanic* passengers, similar to the goal of the Kaggle competition that used this data set (Kaggle Inc. 2012). The project requires students to partake in feature engineering by creating at least one new variable (feature). They are also asked to use a validation scheme and a test set to select and assess the final model. These requirements ensure that students have hands-on experience with these popular data analysis techniques.

In the report, the logistic regression classifier is first described using text and math, before the code and estimates are displayed directly below (Figure 12, which is the same as Figure 3). Further explanation is then provided to justify the model selection. Here, JupyterLab's ability to interweave text, mathematical equations, code, and results into the same document really shines.

Figure 12 A Jupyter Notebook with Narrative Text, Code, and Results



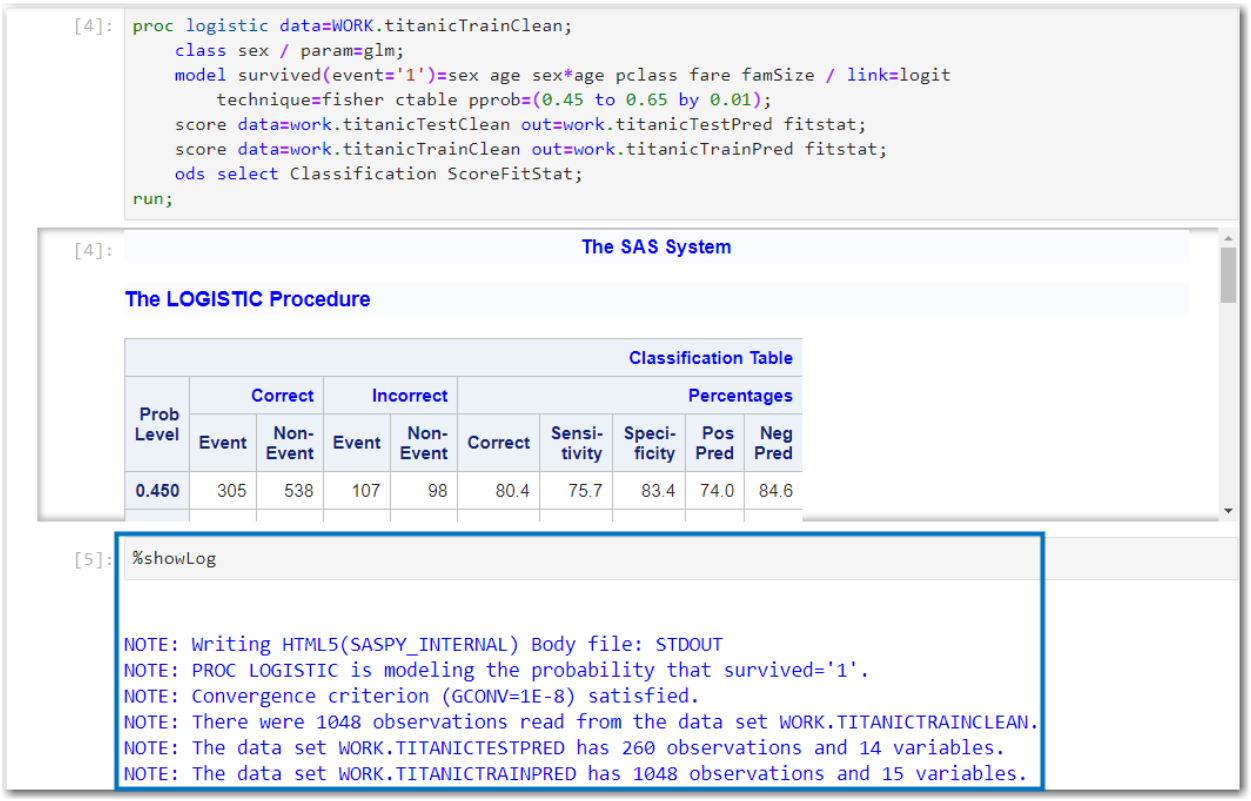
Tips for Using JupyterLab

Now that you have a better understanding of JupyterLab and how it can be used with SAS to teach and learn data science, here are a few additional tips and tricks.

Viewing the SAS Log with Magic Functions

If you execute SAS code that generates results, then those results are displayed below the corresponding code cell unless there are errors. However, experienced SAS users are accustomed to having access to the SAS log and know the importance of checking it for errors and warnings. In a Jupyter notebook, you can accomplish this by using special built-in functions called magic functions, or *magics*. The SAS kernel includes two magics, %showLog and %showFullLog, that you can use to display the log of the previous code submission (Figure 13). Another useful magic is %lsmagic, which lists the magics that are available for the selected kernel.

Figure 13 SAS Log Displayed by Using Magic Functions



```
[4]: proc logistic data=WORK.titanicTrainClean;
         class sex / param=glm;
         model survived(event='1')=sex age sex*age pclass fare famSize / link=logit
             technique=fisher ctable pprob=(0.45 to 0.65 by 0.01);
         score data=work.titanicTestClean out=work.titanicTestPred fitstat;
         score data=work.titanicTrainClean out=work.titanicTrainPred fitstat;
         ods select Classification ScoreFitStat;
     run;
```

[4]: The SAS System

The LOGISTIC Procedure

Classification Table

Prob Level	Correct		Incorrect		Percentages				
	Event	Non-Event	Event	Non-Event	Correct	Sensi-tivity	Speci-ficity	Pos Pred	Neg Pred
0.450	305	538	107	98	80.4	75.7	83.4	74.0	84.6

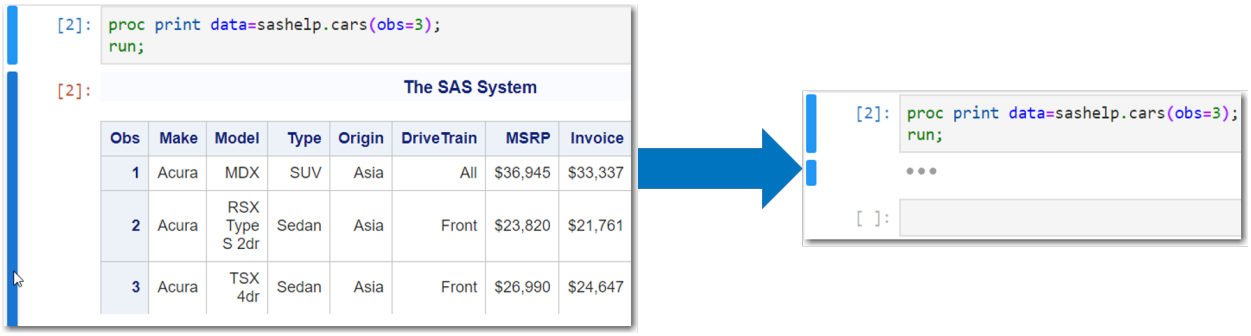
```
[5]: %showLog

NOTE: Writing HTML5(SASPY_INTERNAL) Body file: STDOUT
NOTE: PROC LOGISTIC is modeling the probability that survived='1'.
NOTE: Convergence criterion (GCONV=1E-8) satisfied.
NOTE: There were 1048 observations read from the data set WORK.TITANICTRAINCLEAN.
NOTE: The data set WORK.TITANICTESTPRED has 260 observations and 14 variables.
NOTE: The data set WORK.TITANICTRAINPRED has 1048 observations and 15 variables.
```

Controlling Code Output

The ability to display results directly below the executed code is a helpful feature, but sometimes the amount of output can be overwhelming and can interrupt the flow of the notebook. The notebook editor in JupyterLab includes the following enhancements to address this problem:

- **Collapsible cells**: You can now collapse cells and output by clicking the vertical blue bar to the left of the desired cell (Figure 14). The **View** menu also has options to collapse or expand all code or output cells.

Figure 14 Collapsible Cells



```
[2]: proc print data=sashelp.cars(obs=3);
     run;
```

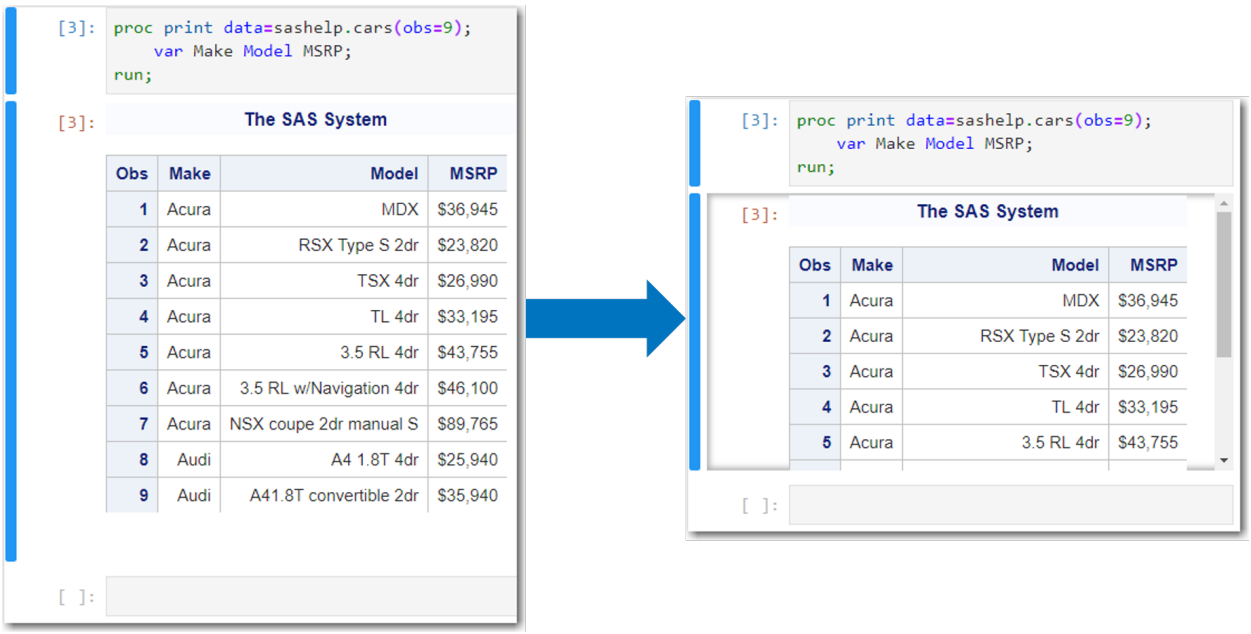
[2]: The SAS System

Obs	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice
1	Acura	MDX	SUV	Asia	All	$36,945	$33,337
2	Acura	RSX Type S 2dr	Sedan	Asia	Front	$23,820	$21,761
3	Acura	TSX 4dr	Sedan	Asia	Front	$26,990	$24,647

```
[2]: proc print data=sashelp.cars(obs=3);
     run;
     •••
[ ]:
```

- **Output scrolling**: You can reduce the space that is allocated to output by enabling scrolling, which shrinks the visible portion of the output and adds a scroll bar to navigate it (Figure 15). You can enable this feature by right-clicking on the output and selecting **Enable Scrolling for Outputs**.

Figure 15 Scrolling in Output Cells



NOTE: For these changes to the output to persist when the notebook is opened again, you must select **Save Notebook with View State** from the **File** menu. Also, these settings are reset if the corresponding code cell is re-executed.

There are also a couple of methods specific to SAS that you can use to fine-tune the code output. The first method is to use ODS statements to display only the desired results. You can do this as follows:

1. Place the code of interest between ODS TRACE ON and ODS TRACE OFF statements. Information is printed to the log about the output objects that were generated by all the code between those statements (Figure 16).
2. Execute the %showLog magic to view the log and discover the names of the output objects of interest (Figure 16).
3. Insert an ODS SELECT statement that contains the names of the desired output objects within the procedure's code (Figure 17).

The second method is borrowed from an example on the SAS kernel project's GitHub page (SAS Institute Inc. 2019b). This method is useful if your SAS code in a cell does not generate any results, in which case the log is displayed as the cell's output. Two system options that are related to controlling the log output are NOSOURCE and NONOTES, which are specified using a global OPTIONS statement. The NOSOURCE option omits the listing of SAS statements in the log unless there are errors. The NONOTES option suppresses system notes such as licensing and site information in addition to the number of observations and variables in a data set. Because these are global options, they remain in effect until you enable them again by specifying the SOURCE and NOTES options.

Figure 16 Output Object Names Identified by ODS TRACE Statements

```
[6]: ods trace on;
     proc logistic data=WORK.titanicTrainClean;
         class sex / param=glm;
         model survived(event='1')=sex age sex*age pclass fare famSize / link=logit
             technique=fisher ctable pprob=(0.45 to 0.65 by 0.01);
         score data=work.titanicTestClean out=work.titanicTestPred fitstat;
         score data=work.titanicTrainClean out=work.titanicTrainPred fitstat;
     run;
     ods trace off;

     •••

[7]: %showLog

     Label:      Classification Table
     Template:   Stat.Logistic.Classification
     Path:       Logistic.Classification
     -------------

     Output Added:
     -------------
     Name:       ScoreFitStat
     Label:      Fit Statistics for SCORE data.
     Template:   Stat.Logistic.ScoreFitStat
     Path:       Logistic.ScoreFitStat
```

Figure 17 Output Selected by the ODS SELECT Statement

```
[8]: proc logistic data=WORK.titanicTrainClean;
         class sex / param=glm;
         model survived(event='1')=sex age sex*age pclass fare famSize / link=logit
             technique=fisher ctable pprob=(0.45 to 0.65 by 0.01);
         score data=work.titanicTestClean out=work.titanicTestPred fitstat;
         score data=work.titanicTrainClean out=work.titanicTrainPred fitstat;
         ods select ScoreFitStat;
     run;
```

[8]:

The SAS System

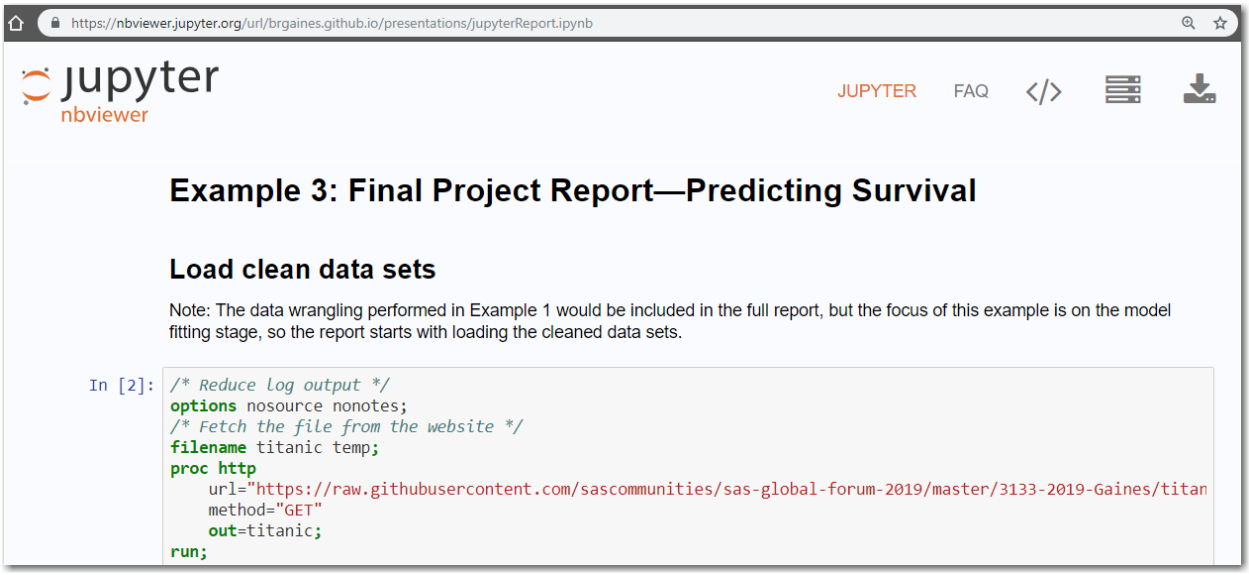
The LOGISTIC Procedure

Data Set	Total Frequency	Log Likelihood	Error Rate	AIC	AICC	BIC	SC	R-Square
WORK.TITANICTESTCLEAN	260	-132.0	0.2308	278.0976	278.5421	303.0224	303.0224	0.263161
WORK.TITANICTRAINCLEAN	1048	-463.2	0.1861	940.4222	940.5299	975.1046	975.1046	0.361413

Sharing Notebooks

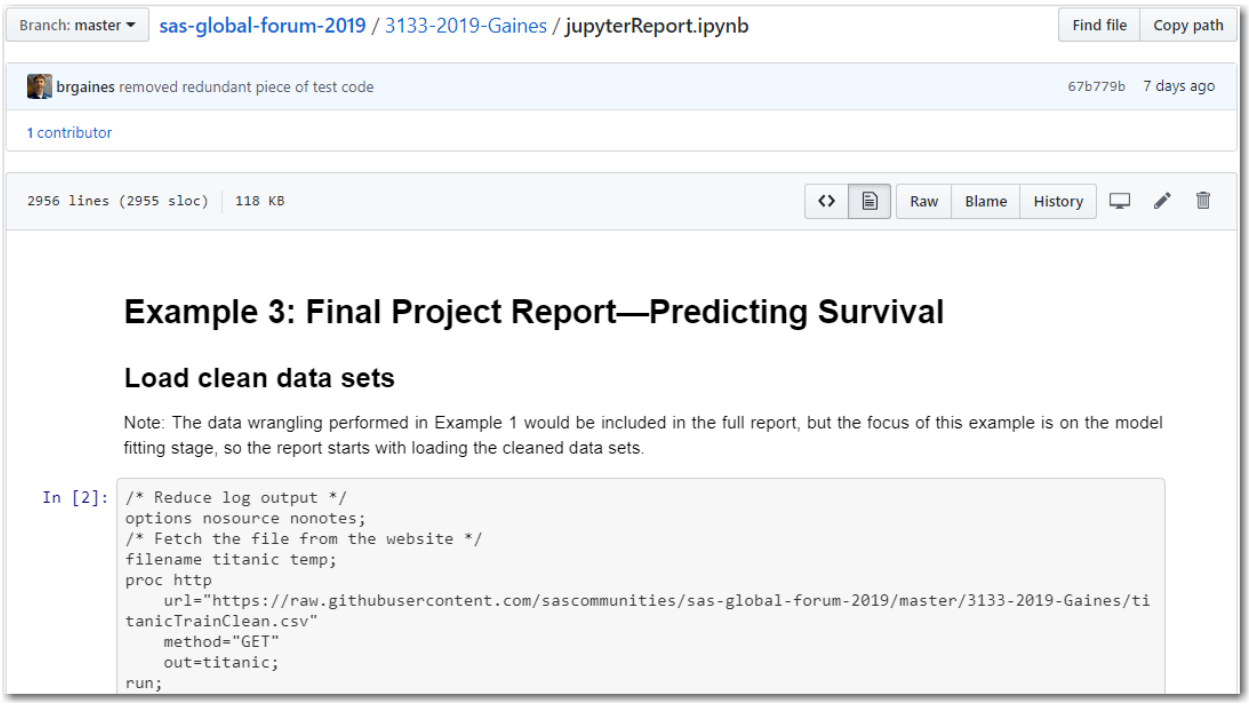
You can share Jupyter notebooks just as you would any other file by using email or a cloud file storage service, although you should also include supporting files such as data sets. Exporting the notebook to HTML alleviates the need for the recipient to view the notebook in a Jupyter interface, and the HTML version can also be readily shared on a website. If the notebook file itself is uploaded to a website, then you can also view it by entering the URL into the Jupyter Notebook Viewer (`https://nbviewer.jupyter.org/`), as shown in Figure 18 and at `http://nbviewer.jupyter.org/url/brgaines.github.io/presentations/jupyterReport.ipynb`.

Figure 18 Jupyter Notebook Viewer



Another option is to use Git, a version-control system that enables you and your collaborators to manage and track changes to the files in a project. If the collection of files, known as a *repository*, is hosted remotely using a service such as GitHub or GitLab, then the notebook is automatically rendered when the remote repository is viewed from the hosting service's website (Figure 19). Not only does this eliminate the need to export the notebook to another format before sharing it, but knowledge of a version-control system is itself a useful skill for a data scientist to learn. In fact, it has become increasingly common for instructors to use Git repositories for homework submissions (Çetinkaya-Rundel and Rundel 2018; Hicks and Irizarry 2018). Experimental Git functionality is included in SAS Studio 3.8 and SAS University Edition; for more information, see SAS Institute Inc. (2018a).

Figure 19 Jupyter Notebook Rendered on GitHub



Note-Taking with Markdown

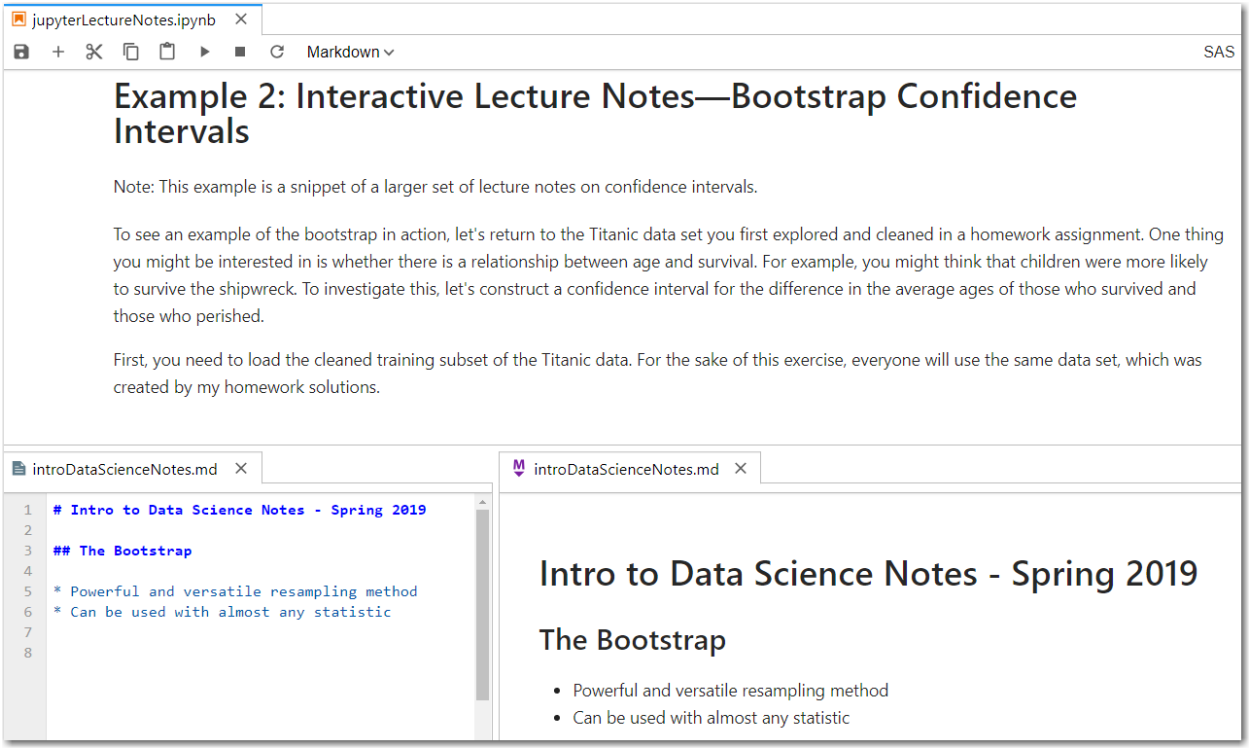
As discussed in the "Jupyter Notebooks" section, Markdown syntax is used within Markdown cells to include explanatory text and mathematical equations in your notebook. A new and useful feature in JupyterLab is the ability to render and preview stand-alone Markdown documents, which have a file extension of .md, by using JupyterLab's text editor. This enables you to take notes during a lecture, or about a notebook or project that you are working on, without having to leave JupyterLab (Figure 20). Using Markdown for note-taking is attractive because it strikes a balance between simplicity and functionality.

You can preview Markdown documents in JupyterLab by following these steps:

1. From the **Launcher** tab, create a Text File to open the text editor.
2. From the **File** menu, select **Save File As** and change the file extension from ".txt" to ".md".
3. Within the text editor, right-click and select **Show Markdown Preview** to open the Markdown preview tab.

Now, after text is added to the Markdown document, the preview tab will update with the rendered formatting (Figure 20).

Figure 20 Live Markdown Preview



Summary

Data science skills are in high demand, and SAS is helping to close the skills gap. SAS University Edition, which includes the popular open-source interface JupyterLab, is one effort in this regard. SAS University Edition enables you to learn valuable SAS analytics skills, and the JupyterLab interface enables you to practice other important data science skills, such as communication skills and reproducibility. Jupyter notebooks provide a flexible environment in which you can efficiently combine narrative text and mathematical notation with code and results in the same document. Assignments, lecture notes, and analytical reports are a few of the ways in which Jupyter notebooks can be used to learn data science, but the possibilities are endless. Together, SAS and JupyterLab are a great combination for learning and teaching data science.

REFERENCES

Baker, M. (2016). "1,500 Scientists Lift the Lid on Reproducibility." *Nature* 533:452–454. https://doi.org/10.1038/533452a.

Baumer, B., Çetinkaya-Rundel, M., Bray, A., Loi, L., and Horton, N. J. (2014). "R Markdown: Integrating a Reproducible Analysis Tool into Introductory Statistics." *Technology Innovations in Statistics Education* 8:22 pages.

Çetinkaya-Rundel, M., and Rundel, C. (2018). "Infrastructure and Tools for Teaching Computing throughout the Statistical Curriculum." *American Statistician* 72:58–65.

Chance, B., Mendoza, S., and Tintle, N. (2018). "Student Gains in Conceptual Understanding in Introductory Statistics with and without a Curriculum Focused on Simulation-Based Inference." In *Proceedings of the International Conference on Teaching Statistics*, vol. 10. The Hague: International Association for Statistical Education.

De Veaux, R. D., Agarwal, M., Averett, M., Baumer, B. S., Bray, A., Bressoud, T. C., Bryant, L., et al. (2017). "Curriculum Guidelines for Undergraduate Programs in Data Science." *Annual Review of Statistics and Its Application* 4:15–30.

Deloitte (2016). "Analytics Trends 2016." Accessed March 1, 2019. https://www2.deloitte.com/us/en/pages/deloitte-analytics/articles/analytics-trends.html.

Dexter, M., Kiser, K., Peters, A., and Corcoran, C. (2016). "Create Web-Based SAS Reports without Having to Be a Web Developer." In *Proceedings of the SAS Global Forum 2016 Conference*. Cary, NC: SAS Institute Inc. https://support.sas.com/resources/papers/proceedings16/SAS4381-2016.pdf.

Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., and Wenderoth, M. P. (2014). "Active Learning Increases Student Performance in Science, Engineering, and Mathematics." *Proceedings of the National Academy of Sciences* 111:8410–8415.

GAISE College Report ASA Revision Committee (2016). "Guidelines for Assessment and Instruction in Statistics Education (GAISE) College Report 2016." http://www.amstat.org/education/gaise.

Glassdoor (2019). "50 Best Jobs in America for 2019." Accessed March 1, 2019. https://www.glassdoor.com/List/Best-Jobs-in-America-LST_KQ0,20.htm.

Hardin, J., Hoerl, R., Horton, N. J., Nolan, D., Baumer, B., Hall-Holt, O., Murrell, P., et al. (2015). "Data Science in Statistics Curricula: Preparing Students to 'Think with Data'." *American Statistician* 69:343–353.

Hemedinger, C. (2017). "Introducing SASPy: Use Python Code to Access SAS." April 8. https://blogs.sas.com/content/sasdummy/2017/04/08/python-to-sas-saspy/.

Hesterberg, T. C. (2015). "What Teachers Should Know about the Bootstrap: Resampling in the Undergraduate Statistics Curriculum." *American Statistician* 69:371–386.

Hicks, S. C., and Irizarry, R. A. (2018). "A Guide to Teaching Data Science." *American Statistician* 72:382–391.

Horton, N. J., Baumer, B. S., and Wickham, H. (2015). "Setting the Stage for Data Science: Integration of Data Management Skills in Introductory and Second Courses in Statistics." *Chance* 28:40–50.

Horton, N. J., and Hardin, J. S. (2015). "Teaching the Next Generation of Statistics Students to 'Think with Data'." *American Statistician* 69:259–265. Guest editorial. Special issue on statistics and the undergraduate curriculum.

Inman, E., and Wright, O. (2017). "Developing Your Own SAS Studio Custom Tasks for Advanced Analytics." In *Proceedings of the SAS Global Forum 2017 Conference*. Cary, NC: SAS Institute Inc. https://support.sas.com/resources/papers/proceedings17/SAS0677-2017.pdf.

Jupyter Team (2015). "Markdown Cells." *The Jupyter Notebook Documentation* .

Kaggle Inc. (2012). "Titanic: Machine Learning from Disaster." Accessed March 1, 2019. https://www.kaggle.com/c/titanic.

McNutt, M. (2014). "Journals Unite for Reproducibility." *Science* 346:679.

Mullis, R. (2018). "Using SAS OnDemand for Academics: Ten Tips for Success." In *Proceedings of the SAS Global Forum 2018 Conference*. Cary, NC: SAS Institute Inc. https://support.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2018/1947-2018.pdf.

Press, G. (2016). "Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says." *Forbes* Accessed March 1, 2019. https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/#559a8ad6f637.

SAS Institute Inc. (2018a). *SAS Studio 3.8: User's Guide*. Cary, NC: SAS Institute Inc.

SAS Institute Inc. (2018b). *SAS Studio: Developer's Guide to Writing Custom Tasks*. Cary, NC: SAS Institute Inc. http://support.sas.com/software/products/sas-studio/.

SAS Institute Inc. (2018c). *SAS/STAT 15.1 User's Guide*. Cary, NC: SAS Institute Inc. http://go.documentation.sas.com/?docsetId=statug&docsetTarget=titlepage.htm&docsetVersion=15.1&locale=en.

SAS Institute Inc. (2019a). "A Python Interface to MVA SAS." Accessed March 1, 2019. http://github.com/sassoftware/saspy.

SAS Institute Inc. (2019b). "SAS Kernel for Jupyter." Accessed March 1, 2019. http://github.com/sassoftware/sas_kernel.

Tintle, N., Clark, J., Fischer, K., Chance, B., Cobb, G., Roy, S., Swanson, T., and VanderStoep, J. (2018). "Assessing the Association between Precourse Metrics of Student Preparation and Student Performance in Introductory Statistics: Results from Early Data on Simulation-Based Inference vs. Nonsimulation-Based Inference." *Journal of Statistics Education* 26:103–109.

U.S. News and World Report (2019). "The 100 Best Jobs in America." Accessed March 1, 2019. https://money.usnews.com/careers/best-jobs/rankings/the-100-best-jobs.

Wicklin, R. (2018). "The Essential Guide to Bootstrapping in SAS." December 12. https://blogs.sas.com/content/iml/2018/12/12/essential-guide-bootstrapping-sas.html.

Acknowledgments

The author is grateful to Ed Huddleston at SAS Institute Inc. for his valuable editorial assistance in the preparation of this paper.

Contact Information

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

Brian R. Gaines
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Brian.Gaines@sas.com