Paper SAS0289-2017

The Well-Equipped Student: Using SAS® University Edition and E-Learning to Gain SAS® Skills

Randy Mullis, Allison Mahaffey, and Parth Patel, SAS Institute Inc.

ABSTRACT

SAS® programming skills are much in-demand, and numerous free tools are available for students who want to develop those skills. This paper introduces students to SAS® Studio and the Jupyter Notebook interface within SAS® University Edition. To make this introduction more tangible, the paper uses a large data set of baseball statistics as an example. In particular, statistical analysis using SAS Studio examines the relationship between salary and performance for major leaguers.

From importing text files to creating basic statistics to doing a more advanced analysis, this paper shows multiple ways to carry out tasks so that you can choose whichever method works best for you. Additional statistics that use t tests and linear regression are simple with SAS University Edition. For completeness, the paper shows the same code that is used in SAS Studio examples in the context of Jupyter Notebook in SAS University Edition.

The paper also provides additional information about the SAS® e-Learning Library and SAS certification to show students how to be fully equipped in order to apply themselves to analytics and data exploration.

INTRODUCTION

Not long ago, well-equipped students entering the job market would need only a degree on their resume. Now, certifications and other skills developed outside of a degree program are becoming essential. Market demands for SAS skills are strong, and the areas of analytics, data science, and big data continue to grow (Renzulli 2016). Students who want a competitive advantage are wise to include using SAS® software as part of their education. Fortunately, SAS provides learning tools at no cost for individuals who are looking to build these valuable analytical skills. These free tools include SAS University Edition, which bundles access to both SAS Studio and Jupyter Notebook. SAS e-Learning courses introduce you to basic programming and statistics skills. Finally, online tutorials show you how to use SAS Studio for data analysis.

This paper provides an overview of these free learning resources. If you are a student or someone just learning SAS, you will learn about the tools available to help you become adept at using SAS. If you are a longtime SAS user who is responsible for teaching others, you will discover some of the ways you can help new users achieve proficiency.

To make the overview more fun and to show the capabilities of SAS software, this paper uses baseball data and SAS programming to calculate basic statistics and display related graphs. Through examples, the paper shows how to use the task-based interface, which requires no coding. Code examples demonstrate that both beginners and experts can use the software.

SAS UNIVERSITY EDITION

One of the easiest ways to learn SAS is by using SAS University Edition. To better understand the benefits of using SAS University Edition, this section puts it in the context of its SAS product ancestors. Initially, SAS was run primarily on a mainframe computer, in a client/server model. Later, SAS was run on a single-user PC, which involved an installation process. In both of these scenarios, you were required to have a SAS license, even if you were just beginning to learn. SAS University Edition, on the other hand, is available for free download, and the latest version of SAS is already installed on it. Because SAS University Edition is a virtual machine, it does not need Internet access, which makes the software mobile and convenient. Moreover, it is comparatively simple to install SAS University Edition. Here are the steps:

1. Install the free virtualization software (Oracle VirtualBox or VMware Workstation Player 12).
2. Download SAS University Edition from SAS.

3. Import the virtual machine that you downloaded from SAS.

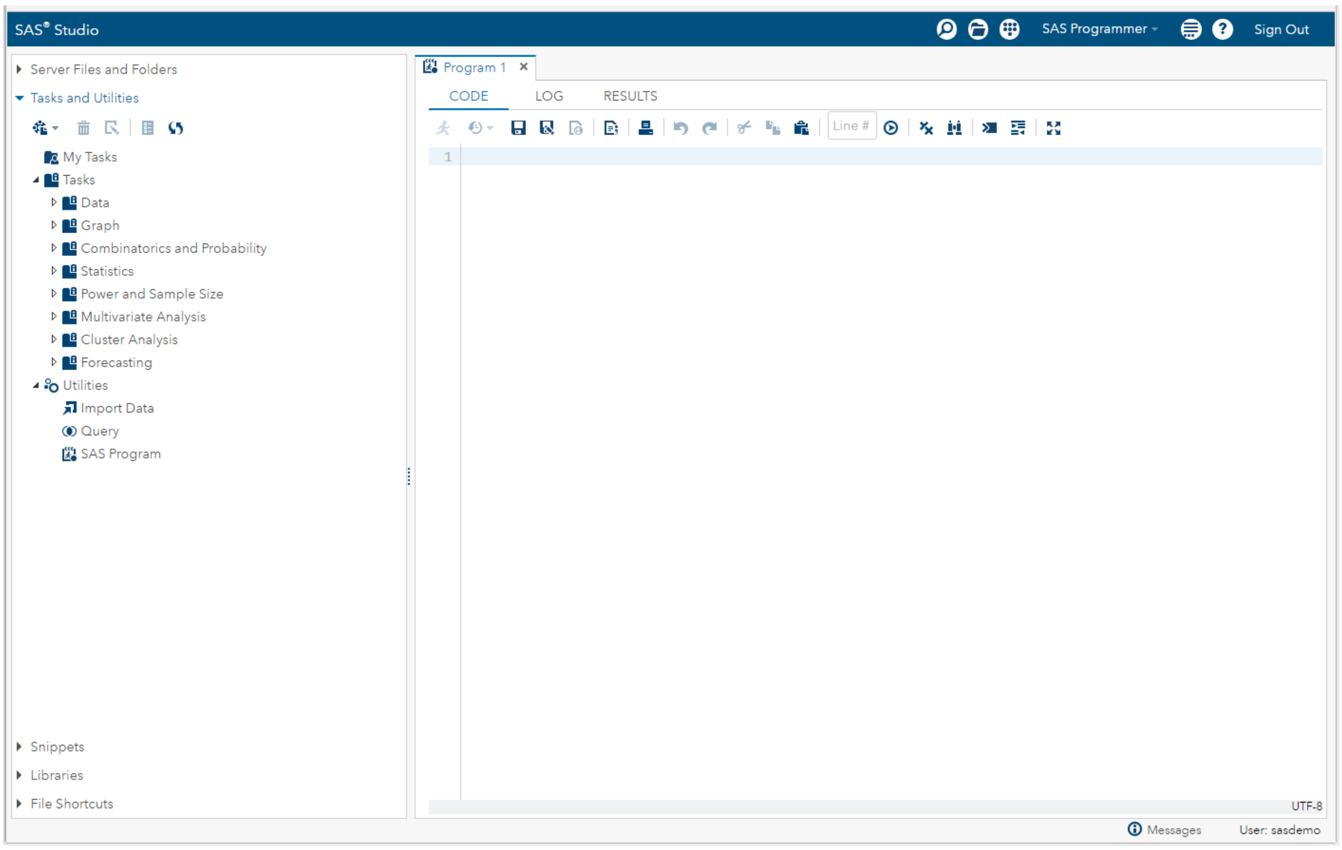
As you can see, it has never been easier to use SAS software! SAS University Edition includes the following SAS products: SAS Studio, Base SAS®, SAS/STAT®, SAS/IML®, SAS/ACCESS® Interface to PC Files, and several time-series forecasting procedures from SAS/ETS®. This free product bundles access to both the SAS Studio and Jupyter Notebook web applications, offering you the flexibility to use whichever interface you prefer to perform your calculations and generate visualizations.

The majority of SAS University Edition users choose SAS Studio to conduct their analyses, either through its tasks and utilities, or by writing SAS code. You access the SAS Studio interface via your web browser, and it includes predefined tasks, code snippets, and an option to enable autocomplete in the editor. With SAS Studio, importing data, joining tables, calculating basic statistics, creating and using SAS macros, and calculating additional statistics and generating visuals is easy.

If you need SAS University Edition installation and configuration tips, refer to the links under the SAS University Edition heading in the Resources section of the paper.

USING TASKS AND UTILITIES

As shown in Display 1 below, SAS Studio contains a number of tasks and utilities in the navigation pane on the left of the screen. You can use these tasks and utilities as they are, or you can change the default settings to customize them. The point-and-click interface is easy to use, and SAS Studio provides prompts to help you because the tasks and utilities are actually wizards that execute actions. To use any of the tasks, browse from among the categories and select the one that meets your needs.



Display 1. Tasks and Utilities in the Navigation Pane

When you select and use a task or utility, a **Split** view opens in the workspace on the right of the screen. This view shows the settings, the code, and the results for the utility or task. For most of the tasks and utilities in SAS Studio, there are three views available: **Settings**, **Code/Results**, and **Split**.

One of the valuable aspects of this interface design is consistency. When you select a task or utility, you expect that it has settings, most with default values. You must ensure that the value for each setting meets your requirements.

Note: Some tasks and utilities contain a large number of settings, so you might need to scroll through more than one screen to see all of the values for each setting.

Another consistent element of the tasks and utilities interface is the presence of buttons that execute standard operations. For example, the workspace contains a running-man icon ( ) that you can click to run the task or utility, using the current settings. The workspace also contains icons for the following actions: **Save**, **Save As**, **Show/Hide** the log, and **Show/Hide** the code. Place your pointer over each icon for a tooltip if you do not know what it does.

Finally, and perhaps most pertinent to learning SAS, you can always see the code generated by the task or utility that you are using in SAS Studio. If your long-term goal is to learn SAS coding, you can begin by using the point-and-click interface of the tasks and utilities, and study the code that is used to execute these actions. As you expand your knowledge of SAS coding, you might find it more efficient to write code to run your analyses.

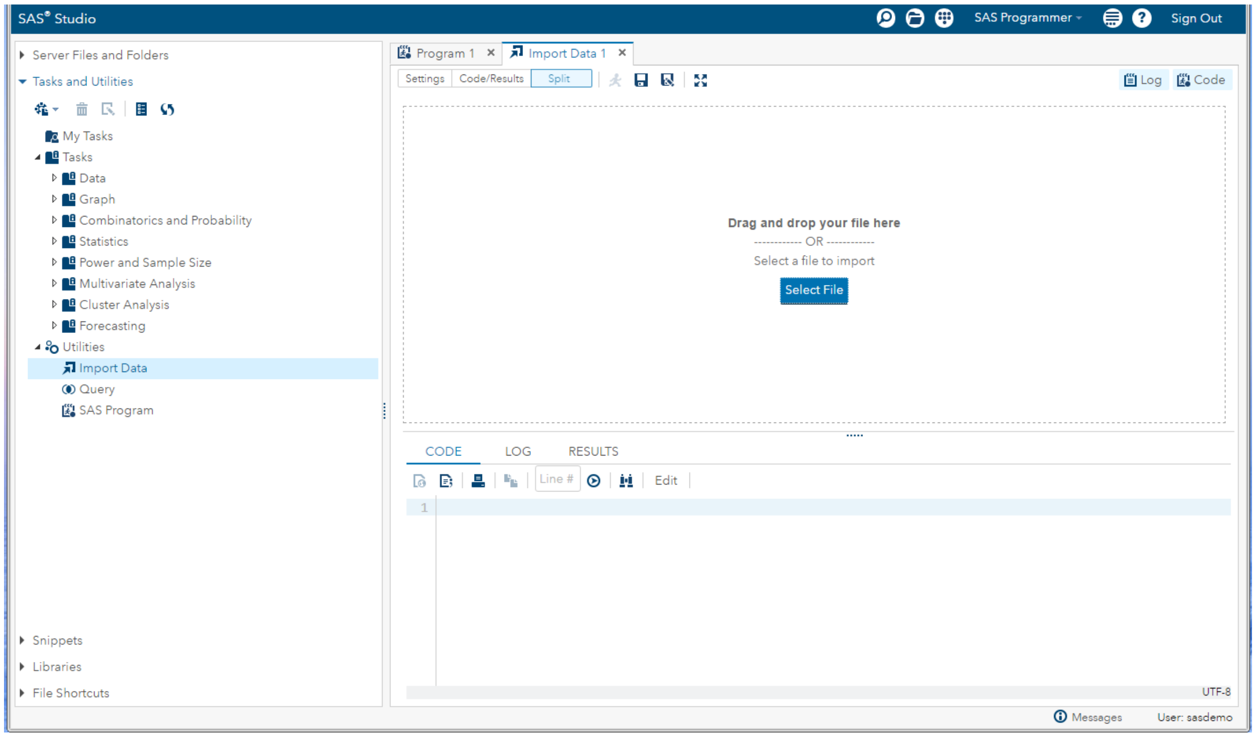
Note: Even when it is not explicitly stated in this paper, you can assume that any of the examples can be performed by using the point-and-click tasks and utilities or by writing code.

For more information about using tasks and utilities, see the SAS University Edition: Help Center. In general, the Help Center contains information about specific topics to help you accomplish what you want in SAS Studio.

IMPORTING DATA

SAS data sets are the native file format for SAS and are efficient for use with SAS procedures. Often, data obtained from sources external to SAS is contained in text files, which are usually in CSV format. Such is the case with the baseball data used for examples in this paper. You must import this CSV data in order to transform it into the SAS file format that is required for data analysis.

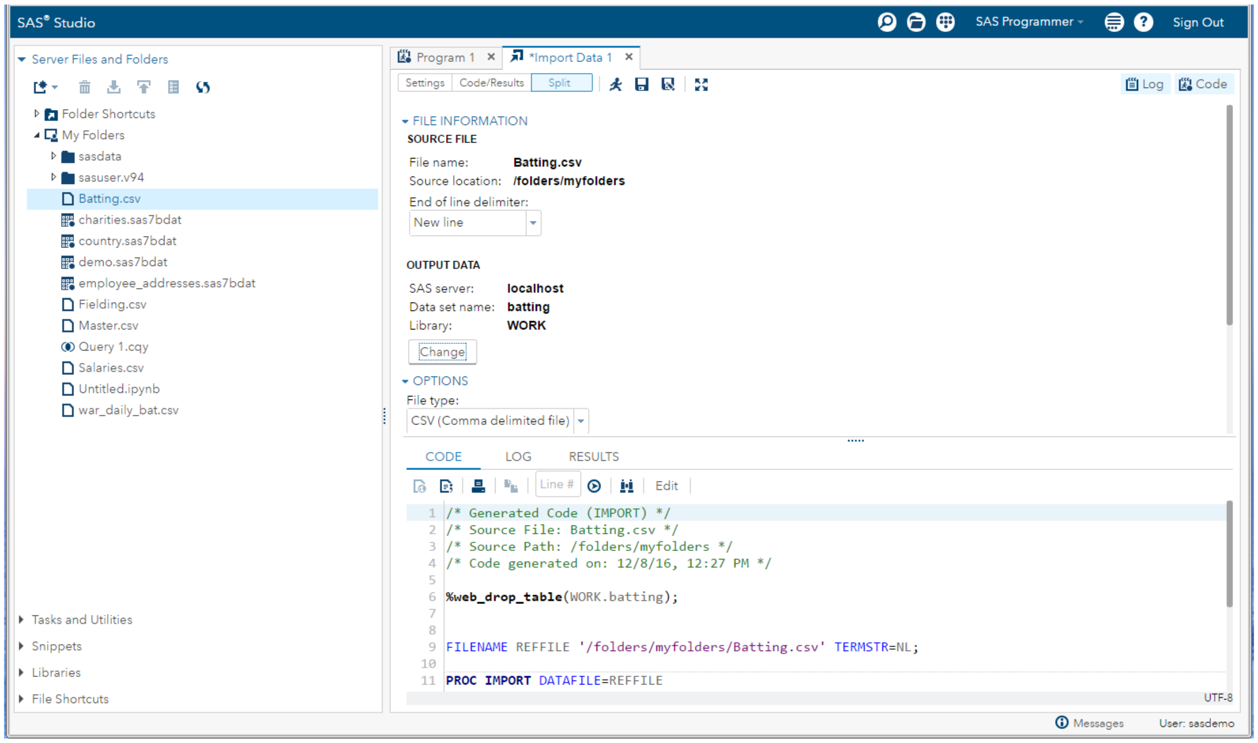
You can import raw data using the **Import Data** utility from the **Tasks and Utilities** section to create native SAS data sets. Select the **Import Data** utility from the **Utilities** section, as shown in Display 2 below. Then, drag your CSV file from the **Server Files and Folders** section.



Display 2. Selecting the Import Data Utility Opens an Import Data Tab in the Workspace

After the **Import Data** tab appears, expand **My folder** in the navigation pane to select the files that you want to import. This example shows the Batting.csv file being imported.

Display 3 below shows the **Split** view that appears in the workspace after you add the file. This view enables you to see both the settings and code for the **Import Data** utility.



FILE INFORMATION

SOURCE FILE

File name: **Batting.csv**
Source location: **/folders/myfolders**
End of line delimiter: New line

OUTPUT DATA

SAS server: **localhost**
Data set name: **batting**
Library: **WORK**

OPTIONS

File type: CSV (Comma delimited file)

```
/* Generated Code (IMPORT) */
/* Source File: Batting.csv */
/* Source Path: /folders/myfolders */
/* Code generated on: 12/8/16, 12:27 PM */

%web_drop_table(WORK.batting);


FILENAME REFFILE '/folders/myfolders/Batting.csv' TERMSTR=NL;

PROC IMPORT DATAFILE=REFFILE
```

Display 3. Import Data Settings (on the Top) and the Corresponding SAS Code (on the Bottom)

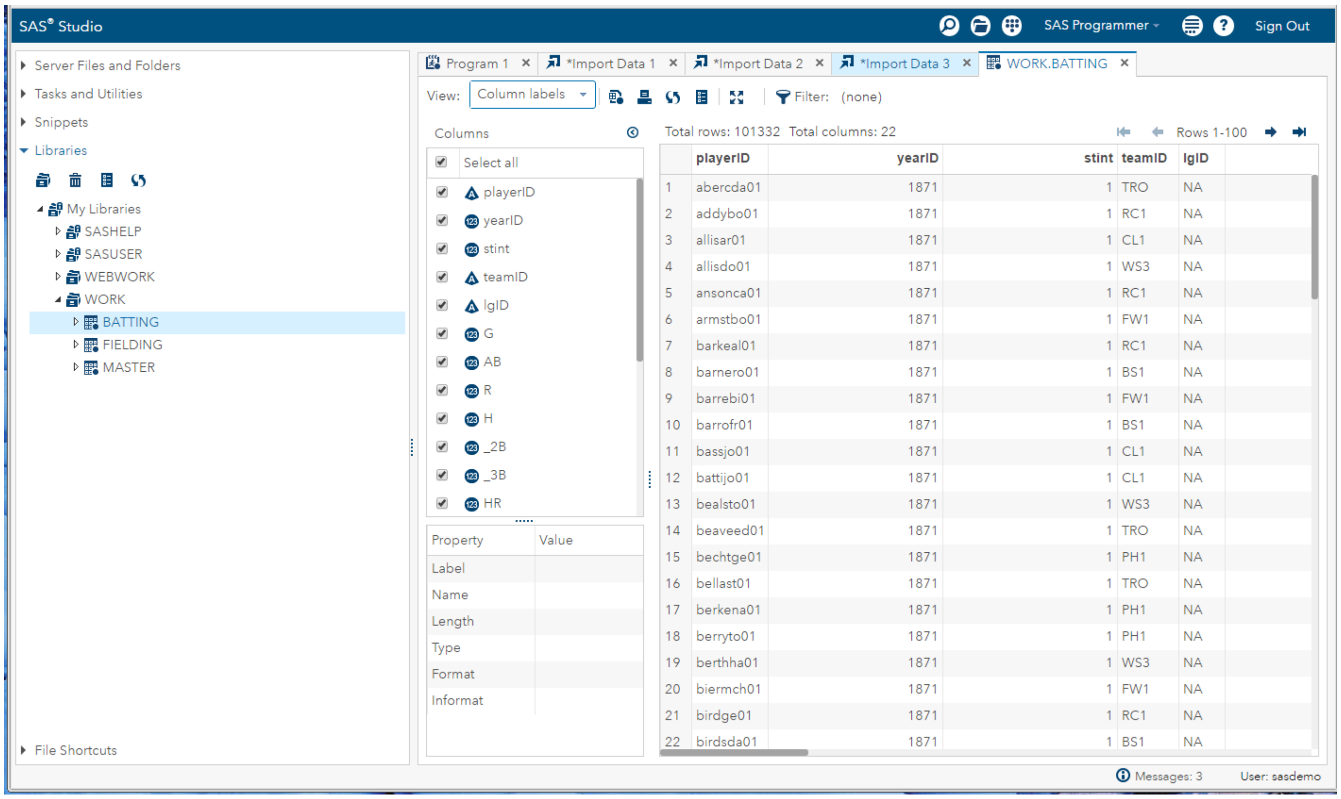
As you can see in Display 3 above, the following changes to **Import Data** values were made because of the particular data files being imported:

End of line delimiter: New Line
Data set name: batting
File type: CSV

These settings (**End of line delimiter**, **Data set name**, and **File type**) are examples of the default settings that you might need to change when using tasks and utilities in SAS Studio. If one of your goals is to learn SAS coding, you can observe how changing these settings is reflected automatically on the **Code** tab. After the changes described above, the code to import the CSV file looks like this:

```
filename reffile '/folders/myfolders/Batting.csv' termstr=nl;
proc import datafile=reffile
    dbms=csv
    out=work.batting;
    getnames=yes;
run;
proc contents data=work.batting;
run;
```

Repeat this process for each data set that you need to import. This example contains data sets for batting and fielding, as well as a master data set that contains player information. To verify that libraries have been created from the CSV files that you imported, go to the **Libraries** section in the navigation pane. Then, double-click the BATTING library to view the data set in SAS Studio as shown below in Display 4.



Total rows: 101332 Total columns: 22

	playerID	yearID	stint	teamID	lgID
1	abercda01	1871	1	TRO	NA
2	addybo01	1871	1	RC1	NA
3	allisar01	1871	1	CL1	NA
4	allisdo01	1871	1	WS3	NA
5	ansonca01	1871	1	RC1	NA
6	armstbo01	1871	1	FW1	NA
7	barkeal01	1871	1	RC1	NA
8	barnero01	1871	1	BS1	NA
9	barrebi01	1871	1	FW1	NA
10	barrofr01	1871	1	BS1	NA
11	bassjo01	1871	1	CL1	NA
12	battijo01	1871	1	CL1	NA
13	bealsto01	1871	1	WS3	NA
14	beaveed01	1871	1	TRO	NA
15	bechtge01	1871	1	PH1	NA
16	bellast01	1871	1	TRO	NA
17	berkena01	1871	1	PH1	NA
18	berryto01	1871	1	PH1	NA
19	berthha01	1871	1	WS3	NA
20	biermch01	1871	1	FW1	NA
21	birdge01	1871	1	RC1	NA
22	birdsda01	1871	1	BS1	NA

Display 4. The BATTING Data Set after Importing the CSV File

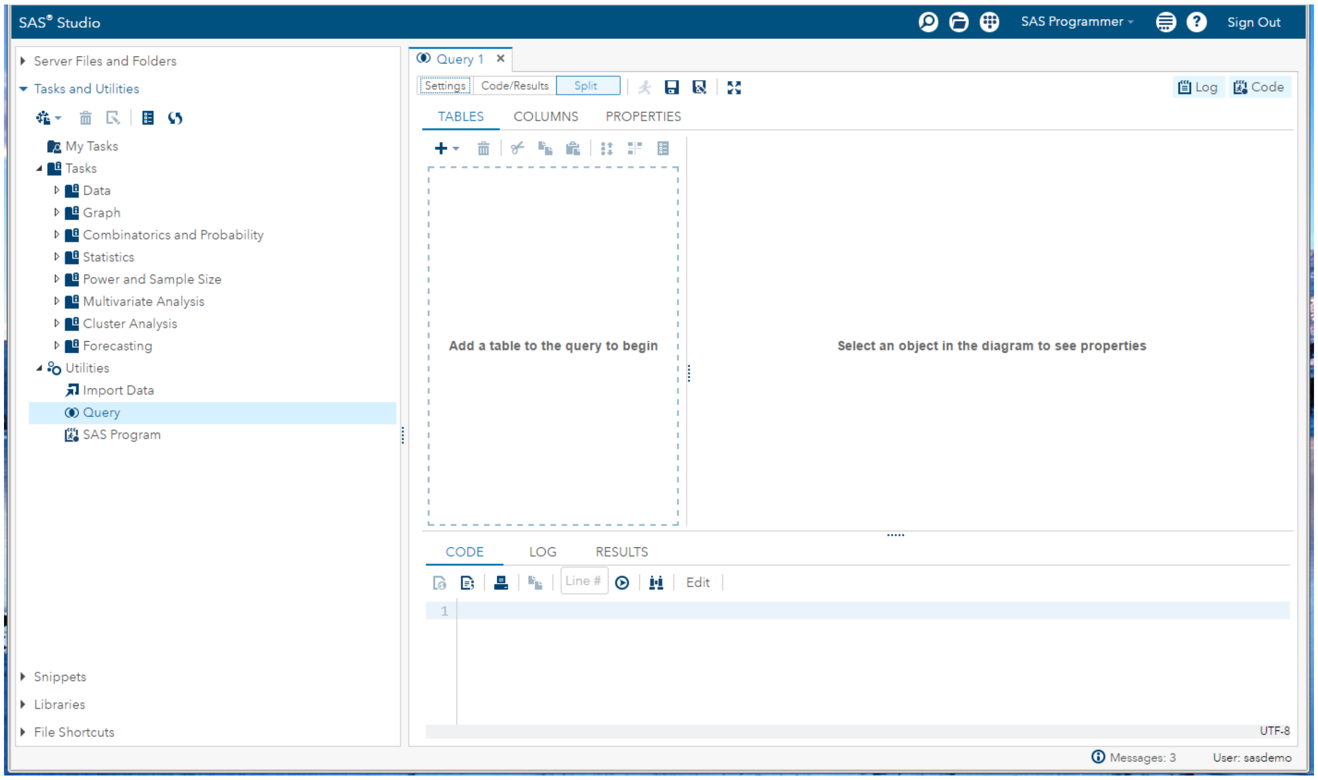
JOINING TABLES

When you are using data sets (usually called tables in the context of SQL queries) for analysis, you typically use columns from each table because of the relationships between them. Rather than merging all the columns into a single table, it is often more efficient to keep the tables separate. In these situations, if the tables contain a common column, you can use this column to join the tables. There are two ways to join tables in SAS: the SQL procedure and the DATA step. It is beyond the scope of this paper to discuss the differences in these approaches, but two papers are listed in the References section that you can read to learn more about it (Foley 2005) (Droogendyk 2008). For the purposes of this paper, PROC SQL is used as an example. You can execute this method using the **Query** utility or code in SAS Studio. After you know how to create code, you might find it more efficient and easier to code your program instead of using the utility.

In this example, the MASTER table contains players' first and last names, whereas the BATTING table contains the players' data that you use for basic calculations. In order to show players' full names alongside your calculations, you must right-join the MASTER and BATTING tables by playerID. You can accomplish this using the **Query** utility or PROC SQL, both of which are described below.

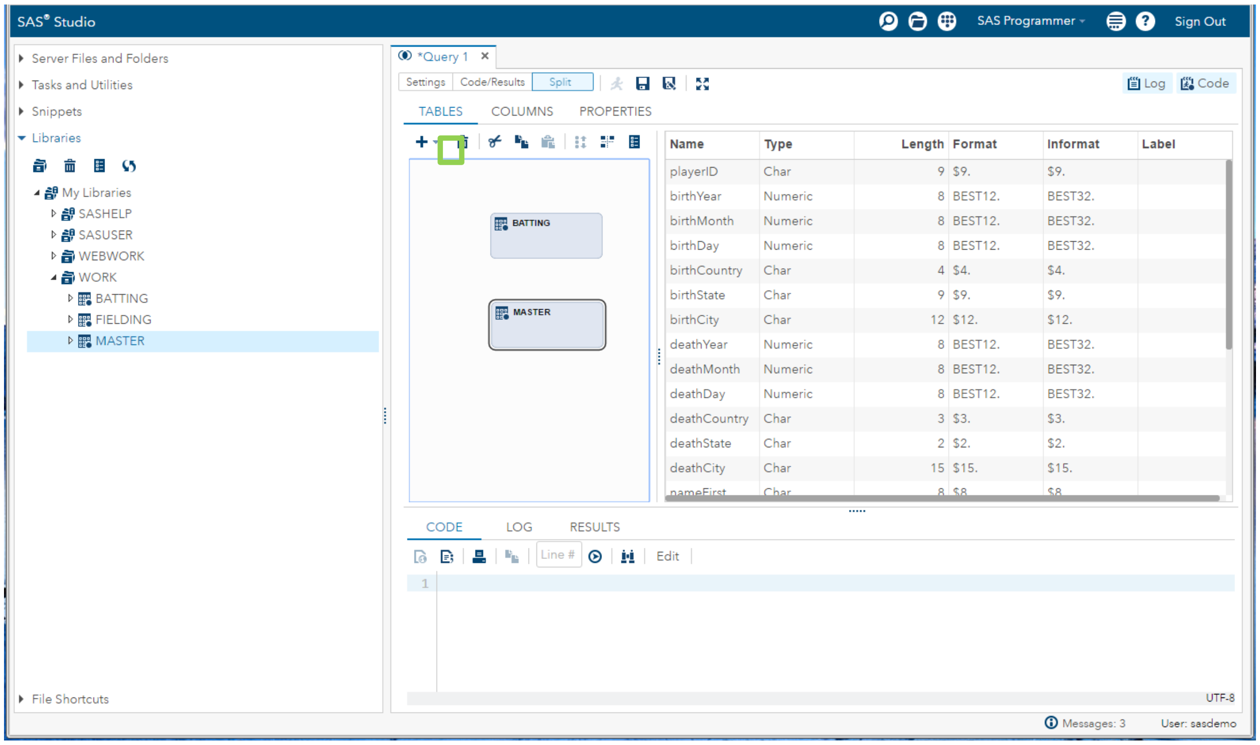
Query Utility

The **Query** utility helps you form structured queries to extract data from a single table or multiple tables based on specific parameters. Using SAS Studio's simple interface, you can drag the BATTING and MASTER tables directly from the **Libraries** section into the **Query** utility. Place them in the box labeled "Add a table to the query to begin" shown below in Display 5.



Display 5. Selecting the Query Utility Shows the Interface for Constructing a Join

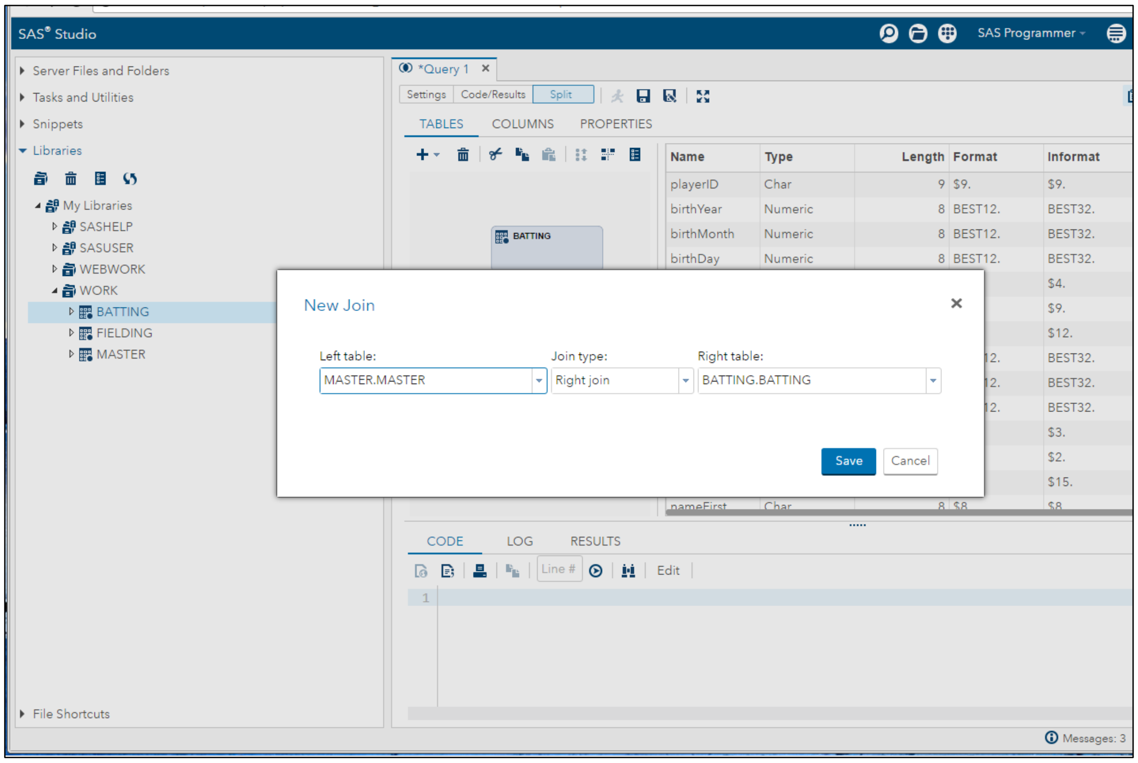
After selecting the tables to join, click the + icon that is highlighted in green in Display 6 below, and then click **Join**.



Name	Type	Length	Format	Informat	Label
playerID	Char	9	$9.	$9.	
birthYear	Numeric	8	BEST12.	BEST32.	
birthMonth	Numeric	8	BEST12.	BEST32.	
birthDay	Numeric	8	BEST12.	BEST32.	
birthCountry	Char	4	$4.	$4.	
birthState	Char	9	$9.	$9.	
birthCity	Char	12	$12.	$12.	
deathYear	Numeric	8	BEST12.	BEST32.	
deathMonth	Numeric	8	BEST12.	BEST32.	
deathDay	Numeric	8	BEST12.	BEST32.	
deathCountry	Char	3	$3.	$3.	
deathState	Char	2	$2.	$2.	
deathCity	Char	15	$15.	$15.	
nameFirst	Char	8	$8	$8	

Display 6. Clicking the + Icon Prompts You to Add a Table or Create a Join

Display 7 shows that MASTER was selected for the left table, and BATTING was selected for the right table. You could also reverse the order of these selections; the type of join—inner, left, right, or full—is the important element to notice here.



New Join

Left table: MASTER.MASTER

Join type: Right join

Right table: BATTING.BATTING

Display 7. A Dialog Box Prompts You to Select the Tables to Join and the Type of Join

The primary value of using this utility is that you do not need to memorize any syntax because the options are available in menu form. When you click **Save**, the query is saved, although it cannot be executed yet because a few more settings are required.

In Display 7 above, the running-man icon (that executes the query) is not active. To run the query, you must use the **Columns** tab to select the columns that you want to include in the resulting joined data set. Use the + icon to add columns from the two tables. By default, the resulting data set is called **Query**. You can change this name on the **Properties** tab. After you have changed these settings, the screen looks like the one in Display 8 below. Running the query by clicking the running-man icon results in a new table that contains columns from each of the two tables joined by playerID.

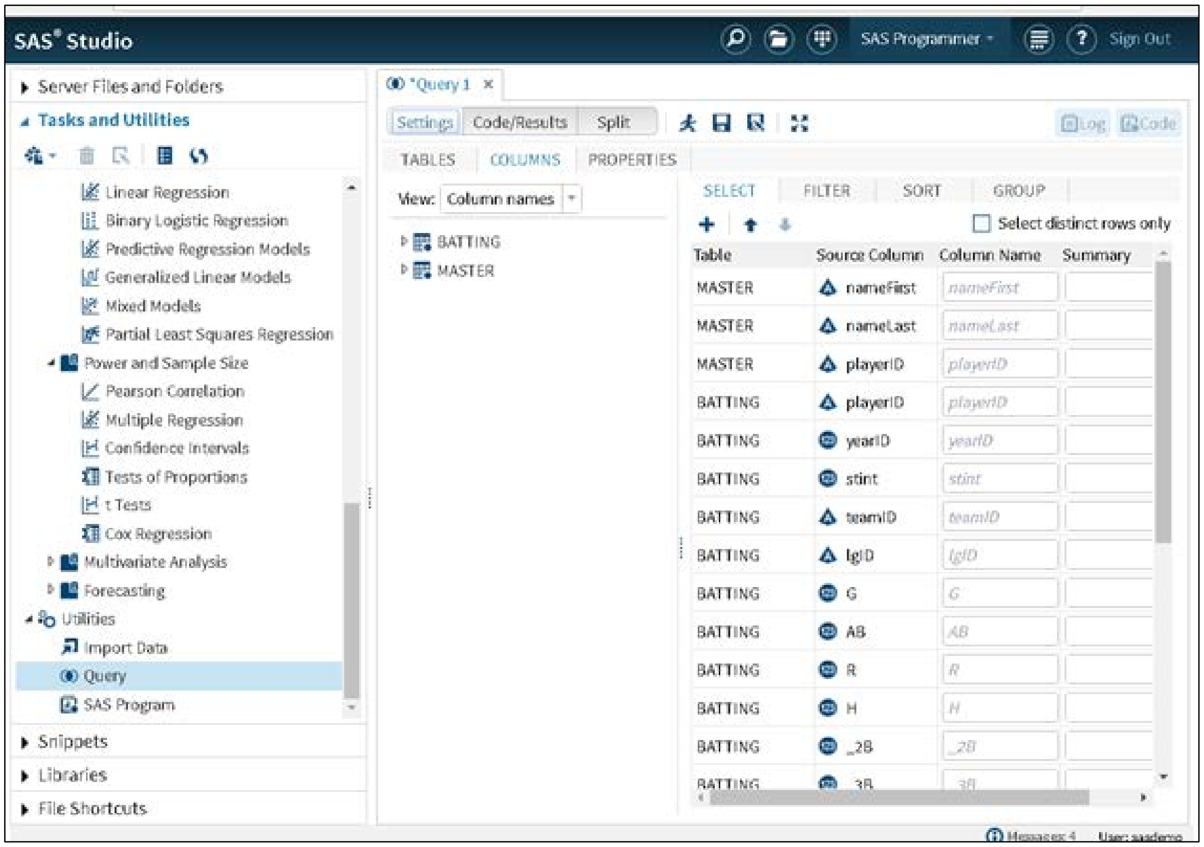


Table	Source Column	Column Name	Summary
MASTER	nameFirst	nameFirst	
MASTER	nameLast	nameLast	
MASTER	playerID	playerID	
BATTING	playerID	playerID	
BATTING	yearID	yearID	
BATTING	stint	stint	
BATTING	teamID	teamID	
BATTING	lgID	lgID	
BATTING	G	G	
BATTING	AB	AB	
BATTING	R	R	
BATTING	H	H	
BATTING	_2B	_2B	
BATTING	3B	3B	

Display 8. Clicking the Running-Man Icon Executes the Query After the Settings Are Changed

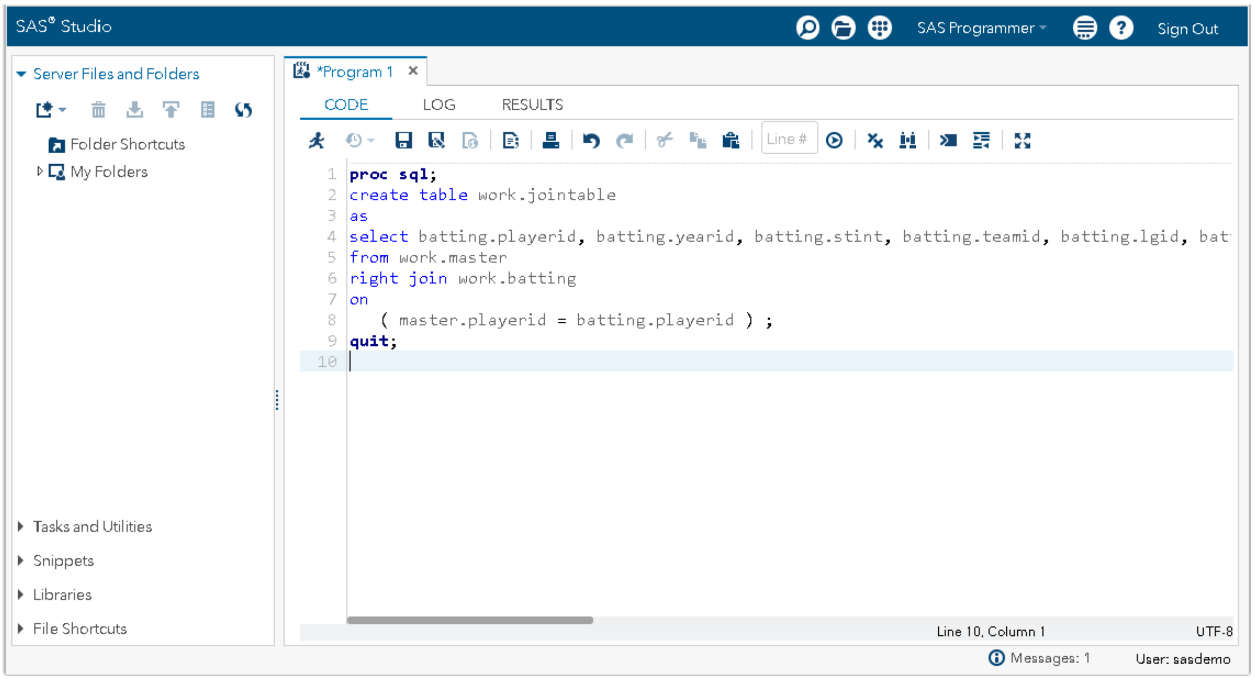
When you use the **Query** utility, you can look at the SAS code that it generates by clicking the **Code** tab. In this way, you can see how the choices you made in the settings are expressed in the SAS code that creates the join. If your objective is to manipulate the data and get work done, you can use the **Query** utility. If your objective is to learn SAS coding, you can create and review well-formed queries using the **Query** utility's point-and-click interface until you are ready to code more directly.

SQL Joins

If you are a more experienced programmer, you can perform these joins by running SQL code such as the following in a basic SAS program:

```
proc sql;
create table work.jointable
as
select batting.playerid, batting.yearid, batting.stint, batting.teamid,
batting.lgid, batting.g, batting.ab, batting.r, batting.h, batting._2b,
batting._3b, batting.hr, batting.rbi, batting.sb, batting.cs, batting.bb,
batting.so, batting.ibb, batting.hbp, batting.sh, batting.sf,
batting.var22, master.namefirst, master.namelast
from work.master
right join work.batting
on
   ( master.playerid = batting.playerid ) ;
quit;
```

You can create this program directly in the SAS Studio code editor, and run it by clicking the running-man icon. Display 9 below illustrates the syntax highlighting that the code editor uses to assist with your coding.



```
proc sql;
create table work.jointable
as
select batting.playerid, batting.yearid, batting.stint, batting.teamid, batting.lgid, bat
from work.master
right join work.batting
on
   ( master.playerid = batting.playerid ) ;
quit;
```

Display 9. Syntax Highlighting in the Editor Assists Coding

CALCULATING BASIC STATISTICS

The first half of this paper has primarily shown how to use SAS Studio tasks and utilities to perform jobs. The second half of the paper focuses more on coding. The purpose of this focus is because, as a student works to gain SAS skills for the purpose of employment, writing code becomes more important. However, the majority of the code samples in the remainder of the paper and most other actions you might need to do can also be accomplished by using tasks and utilities. In fact, you can customize the tasks for your purposes if needed. If you like using a point-and-click interface, see the *SAS® Studio 3.5: Task Reference Guide*.

One of the fundamental constructs in the SAS language is the DATA step. Just as a math student has a calculator for advanced operations but still learns basic math, a student of SAS must learn some basic coding constructs even if she or he uses point-and-click coding alternatives for many tasks. You can learn how to write a DATA step using the materials or courses referenced in this paper. A SAS DATA step takes the joined data set created earlier and produces new variables to store the statistics.

Now it is time to obtain basic statistics for the example baseball data using the DATA step! The downloaded baseball data set contains the important numbers for each player, but does not include critical statistics like batting average (BA), slugging percentage (SP), and other statistics that you would want for each player. Therefore, for each row in the data set (a row for each player), you must create a new variable to store the additional statistics. The following code is written directly in SAS Studio to create these statistics. The code is easy-to-read and generates additional basic information that you can use for further analysis.

```
data work.calculations;
set jointable;
   if yearID ne 2015 then delete;
if AB = 0 or AB = . then delete;
   BA = H/AB;
   _1B = H-_2B-_3B-HR;
   drop _1B;
   SP = (_1B+2*_2B+3*_3B+4*HR)/AB;
   ISO = (_2B+2*_3B+3*HR)/AB;
   SecA = (_1B+2*_2B+3*_3B+4*HR-H+BB+SB-CS)/AB;
run;
```

The code above selects statistics for only year 2015 and deletes everything else from the data set. This code does not produce any output other than a log. When you execute a program like this, you should check the log to make sure that there are no errors. Here is the log for this program:

```
61
62         data work.calculations;
63         set jointable;
64         if yearID ne 2015 then delete;
65         if AB = 0 or AB = . then delete;
66         BA = H/AB;
67         _1B = H-_2B-_3B-HR;
68         drop _1B;
69         SP = (_1B+2*_2B+3*_3B+4*HR)/AB;
70         ISO = (_2B+2*_3B+3*HR)/AB;
71         SecA = (_1B+2*_2B+3*_3B+4*HR-H+BB+SB-CS)/AB;
72         run;

NOTE: There were 101332 observations read from the data set JOINTABLE.
NOTE: The data set WORK.CALCULATIONS has 1028 observations and 28
variables.
NOTE: DATA statement used (Total process time):
      real time           0.31 seconds
      cpu time            0.05 seconds
```

As you can see from the log above, there were no errors. Note also that in the code above, ISO stands for Isolated Power, and SecA stands for Secondary average. SecA is the sum of extra bases gained on hits, walks, and stolen bases, minus the number of times caught stealing.

CREATING AND USING SAS MACROS

SAS includes a macro language. According to Chapter 1 of the *SAS® 9.4 Macro Language: Reference, Fifth Edition*, the SAS macro facility “enables you to assign a name to character strings or groups of SAS programming statements. You can work with the names that you created rather than with the text itself” (SAS 2015). You can use SAS macros to perform tasks as simple as reducing the number of keystrokes to tasks as complex as structuring a program based on macro substitutions. In fact, text substitution is the reason macros exist. They are fundamental to the SAS programming language.

This example illustrates how SAS University Edition makes it easy to learn and use the features of SAS, and also shows how macros can make your programming life easier. The following example macro definition sorts the players on a specified team by age:

```
%macro sortAge(datasetname= , team = );
proc sort data = baseball out = &datasetname;
    by age;
    where year_id = 2015 and team_id = &team;
run;

proc print data = &datasetname;
%mend;
```

With this macro, you can pass the abbreviation for any of the 30 major-league teams as input, and print out a list of that team's players sorted by age:

```
/*List of MLB players on the 2015 Atlanta Braves sorted by age.*/
%sortAge(datasetName = ATLSortedAge, team = "ATL");
```

Based on this list, the youngest player on the Atlanta Braves team was Daniel Castro, and the oldest player was A.J. Pierzynski. Additional examples of the use of macros are included in the sample code for this paper, which is available here.

CALCULATING ADDITIONAL STATISTICS AND GENERATING VISUAL REPRESENTATIONS

To review, so far this paper has shown how to import data and create a SAS data set. It has also demonstrated how to join two tables and generate basic statistics. The following examples go further to perform more advanced calculations and produce visual representations of the data.

Baseball fans often want to rank players. The most common way to do this is with the Wins Above Replacement (WAR) formula.

Wins Above Replacement (WAR)

WAR calculates many statistics to create a single number representing a player's value (FanGraphs 2016). The actual meaning of the WAR value, as the name implies, is twofold:

- If the WAR value is positive, this represents the number of additional wins any given team would have if the specified player replaced an average player on the team.
- If the WAR value is negative, this represents the number of fewer wins any given team would have if the specified player replaced an average player on the team.

Here is the formula to calculate WAR (wikiHow 2015):

1. Compile the statistics for each player into a single number, representing each of the categories of statistics in which you are interested (for example, putouts, on-base percentage, slugging average, and so on).
2. Find the league average for each category.
3. Compare a player's statistics to the average (replacement) player.

You can download the WAR value for all players from the Baseball-Reference site (Baseball-Reference 2016). Import this file using the method described in the Importing Data section. After you have completed the import, you will have data that you can use to explore various questions. For example, it is interesting to examine the career WAR values for prominent players. Here is code written for this paper that calculates the average age for a player when he was at his highest WAR value.

```
data maxwar;
   set sortedbaseball;
   by player_id;
     if first.player_id;
      if year_id >= 1996; *first member of the group then output;
run;
```

```
proc print data = maxwar(obs = 10);
run;

proc means data = maxwar mean maxdec=5;
  var age;
run;
```

This code shows that the maximum WAR value for a player occurred when he was 26.5 years old on average.

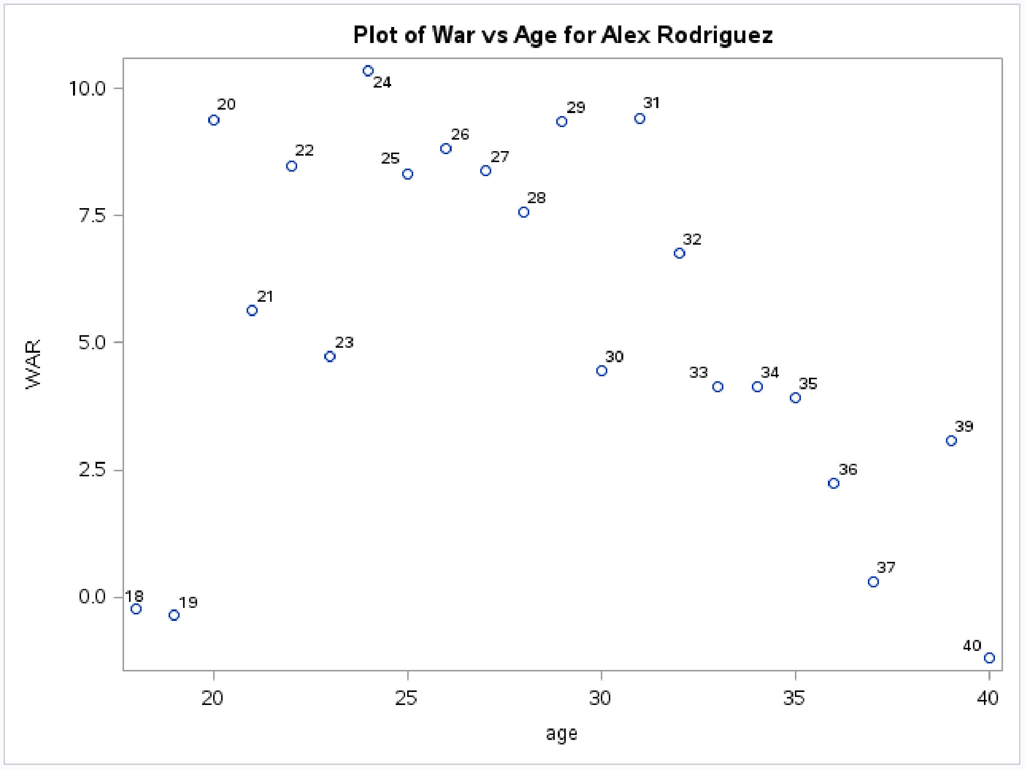
PROC SGPLOT

SAS University Edition includes several SAS products that produce data visualizations. So, you are able to produce meaningful output to help see trends in your data using SAS University Edition just like you would in Base SAS! This example explores how old a player is when he has his highest WAR value. A scatter plot is appropriate for this calculation because it is used to “show how much one variable is affected by another” (Reese 1997). The following scatter plot was made using the SGPLOT procedure, which creates one or more plots and overlays them on a single set of axes (SAS 2016).

The scatter plot in Output 1 shows the prime age for a player in terms of his WAR value. Here is the code:

```
/*WAR vs Age*/
/*Scatter plot for Alex Rodriguez*/
proc sgplot data = baseball;
    scatter x = age y = war/datalabel=age;
    where player_id = "rodrial01";
    title "Plot of WAR vs Age for Alex Rodriguez";
run;
```

Based on the scatter plot shown in Output 1 below, the peak age in terms of WAR for Alex Rodriguez was 24:



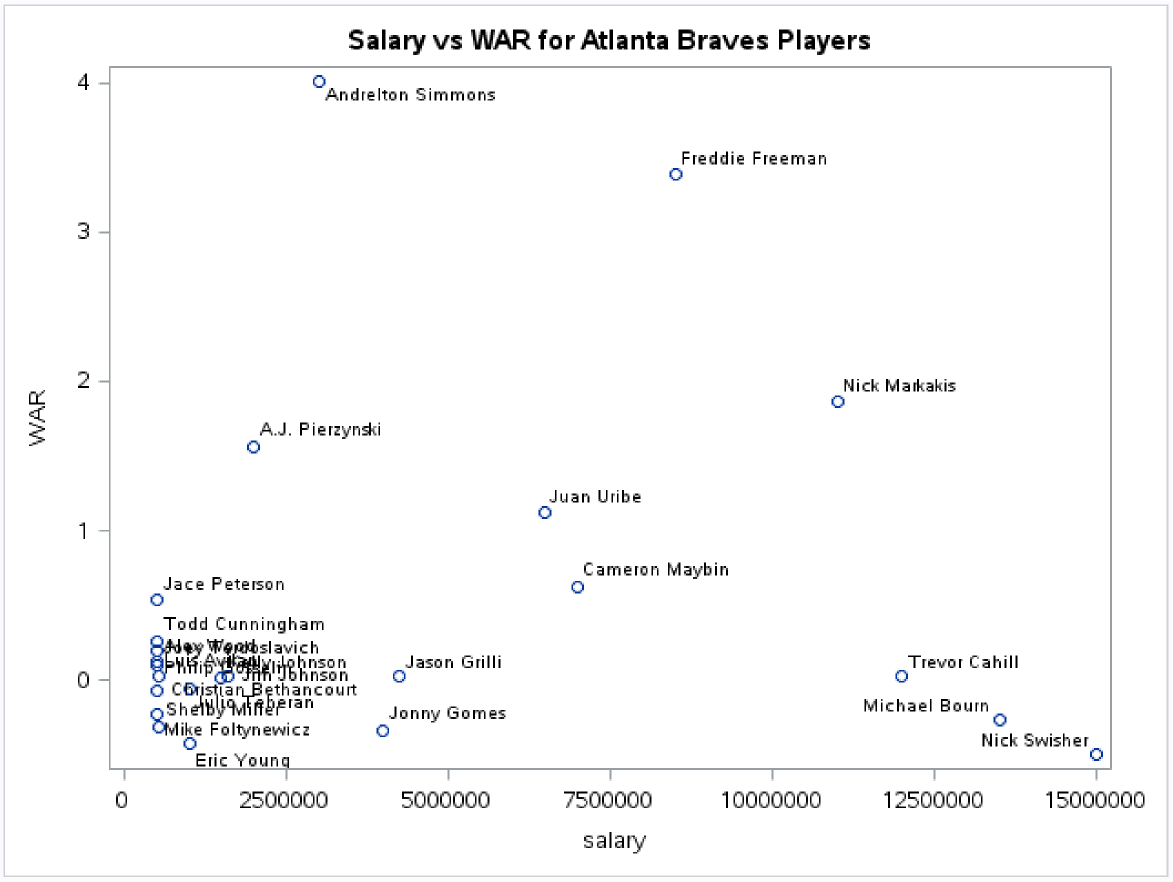
Output 1. Alex Rodriguez’s Highest WAR Relative to Age

A popular topic among baseball fans is the link between on-field performance and salary. You might regard players with a high WAR-to-salary value as providing the best return on the investment. Here is the code to generate a scatter plot showing this analysis:

```
/*B. WAR vs Salary */
data salaryvwar;
   set baseball;
   if year_id = “2015”;
   if salary ~= null;
run;
/*Looking at the salaries of players on the Atlanta Braves compared to
their WAR*/


proc sgplot data = salaryvwar;
   scatter x = salary y = war/datalabel=name_common;
   where team_id = "atl";
   title "Salary vs WAR for Atlanta Braves Players";
run;
```

As you can see in Output 2 below, the Atlanta Braves player with the biggest return on the investment is Andrelton Simmons:



Output 2. The Atlanta Braves Player with the Highest WAR at the Lowest Salary

JUPYTER NOTEBOOK

Beginning in 2016, Jupyter Notebook is included with SAS University Edition. According to the Jupyter site, it is an open-source web application "that allows you to create and share documents that contain live code, equations, visualizations and explanatory text" (Jupyter 2016). Jupyter Notebook supports over 40 programming languages, one of which is SAS. Now you can write SAS programs using the Jupyter Notebook interface, which you might already know. Jupyter Notebook is already used in many academic settings. Instructors sometimes distribute Notebooks to students or require students to turn in Notebooks. So, the ability to create a SAS Notebook with SAS University Edition is very useful.

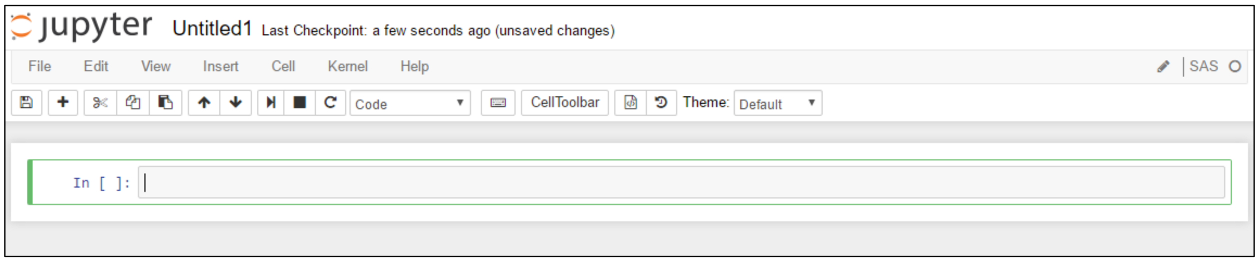
The code used in this section could be developed within Jupyter Notebook or pasted in from another application.

Note: Jupyter Notebook can access the Shared Folder that you created in SAS University Edition because they are both running on the same virtual machine.

To give you some familiarity with Jupyter Notebook, the paper shows a couple of examples of code run in the application below. To access Jupyter Notebook, go to http://localhost:18888 and select **New ► SAS Notebook**.

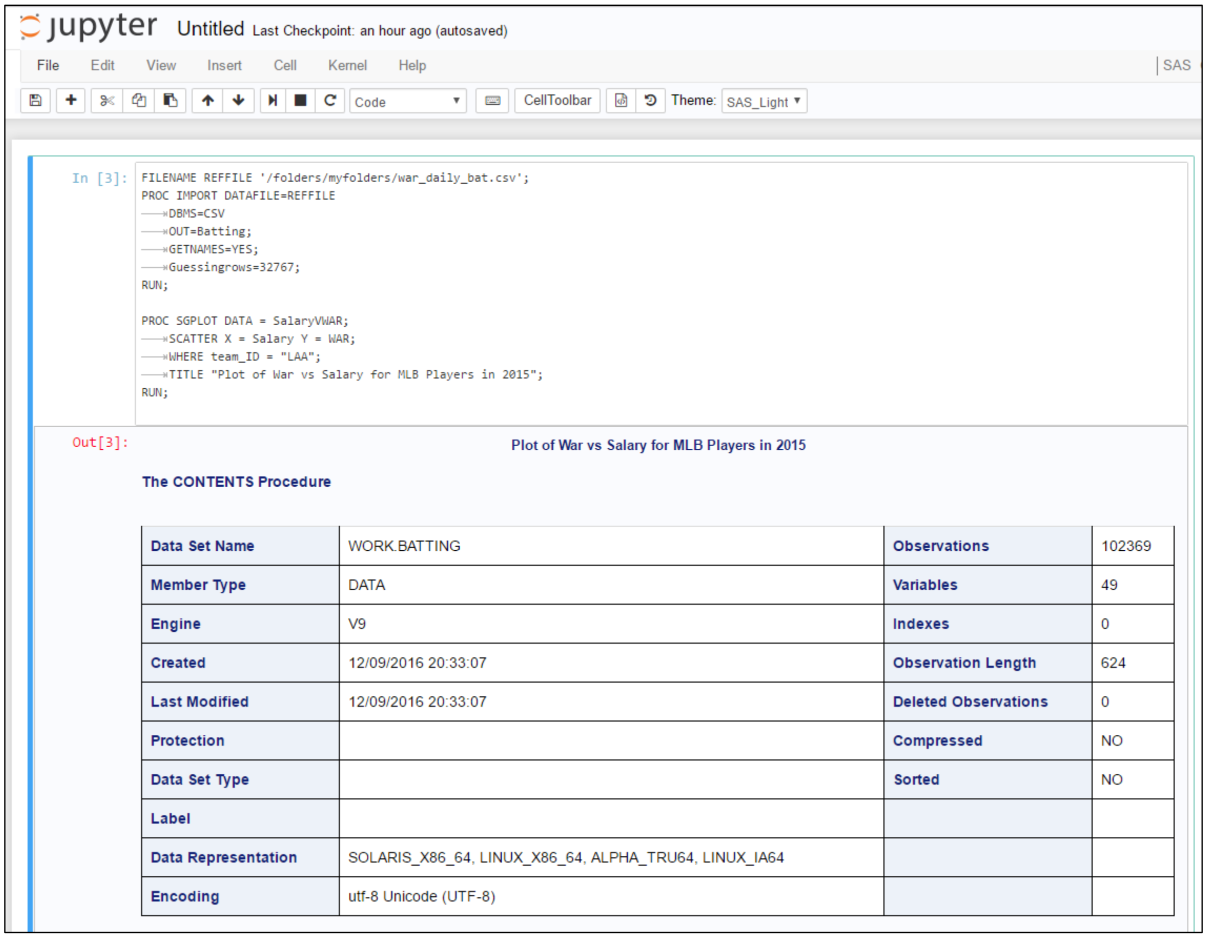
Note: You must have SAS University Edition installed and running when you enter this address in your browser.

Display 10 below shows the input field in which you enter your SAS code from the new SAS Notebook.



Display 10. A New SAS Notebook Containing an Input Field for SAS Code

Run code for WAR versus salary in the Jupyter Notebook. You can see the HTML rich output that Jupyter Notebook produces in Display 11 below:


jupyter Untitled Last Checkpoint: an hour ago (autosaved)
File Edit View Insert Cell Kernel Help
SAS
Code
CellToolbar
Theme: SAS_Light

```
In [3]: FILENAME REFFILE '/folders/myfolders/war_daily_bat.csv';
        PROC IMPORT DATAFILE=REFFILE
            DBMS=CSV
            OUT=Batting;
            GETNAMES=YES;
            Guessingrows=32767;
        RUN;

        PROC SGPLOT DATA = SalaryVWAR;
            SCATTER X = Salary Y = WAR;
            WHERE team_ID = "LAA";
            TITLE "Plot of War vs Salary for MLB Players in 2015";
        RUN;
```

Out[3]:

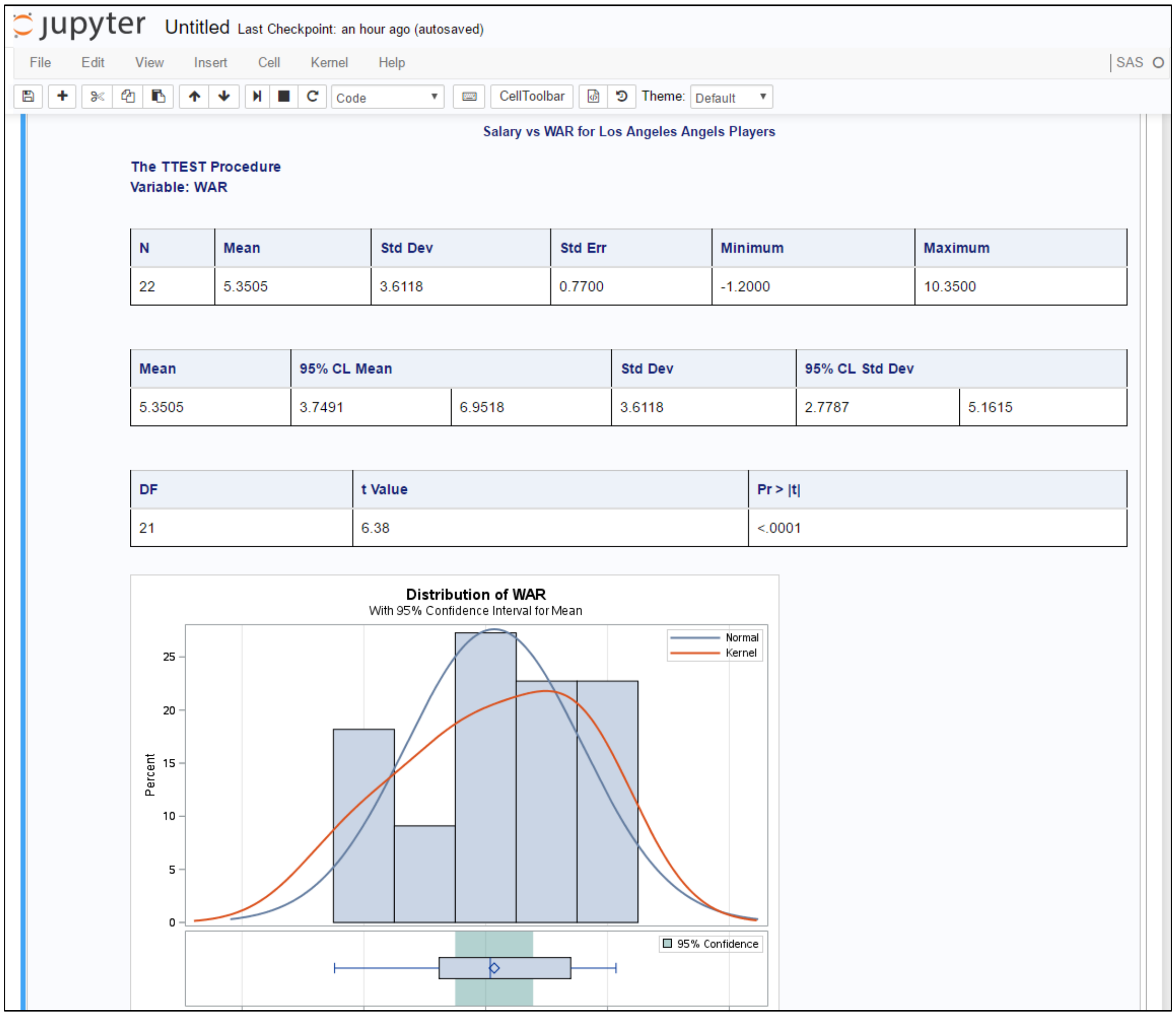
Plot of War vs Salary for MLB Players in 2015

The CONTENTS Procedure

Data Set Name	WORK.BATTING	Observations	102369
Member Type	DATA	Variables	49
Engine	V9	Indexes	0
Created	12/09/2016 20:33:07	Observation Length	624
Last Modified	12/09/2016 20:33:07	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64		
Encoding	utf-8 Unicode (UTF-8)		

Display 11. Output from Executing SAS Code in Jupyter Notebook

Notice that the output from the SAS code in Display 11 above is almost identical to the format of the output you see in SAS Studio. Using SAS in the Jupyter interface, however, means that you can keep a Notebook of SAS code (and add code from other languages if you want) to distribute or to turn in for an assignment. You can give the Notebook to someone, and they can see the results inline. For schools that require Jupyter Notebook, this is a great benefit. Display 12 below shows the rest of the output to further illustrate the similarity of results between the SAS Studio and Jupyter Notebook interfaces.


jupyter Untitled Last Checkpoint: an hour ago (autosaved)
File Edit View Insert Cell Kernel Help
SAS
Code
CellToolbar
Theme: Default

Salary vs WAR for Los Angeles Angels Players

The TTEST Procedure
Variable: WAR

N	Mean	Std Dev	Std Err	Minimum	Maximum
22	5.3505	3.6118	0.7700	-1.2000	10.3500

Mean	95% CL Mean		Std Dev	95% CL Std Dev	
5.3505	3.7491	6.9518	3.6118	2.7787	5.1615

DF	t Value	Pr > |t|
21	6.38	<.0001

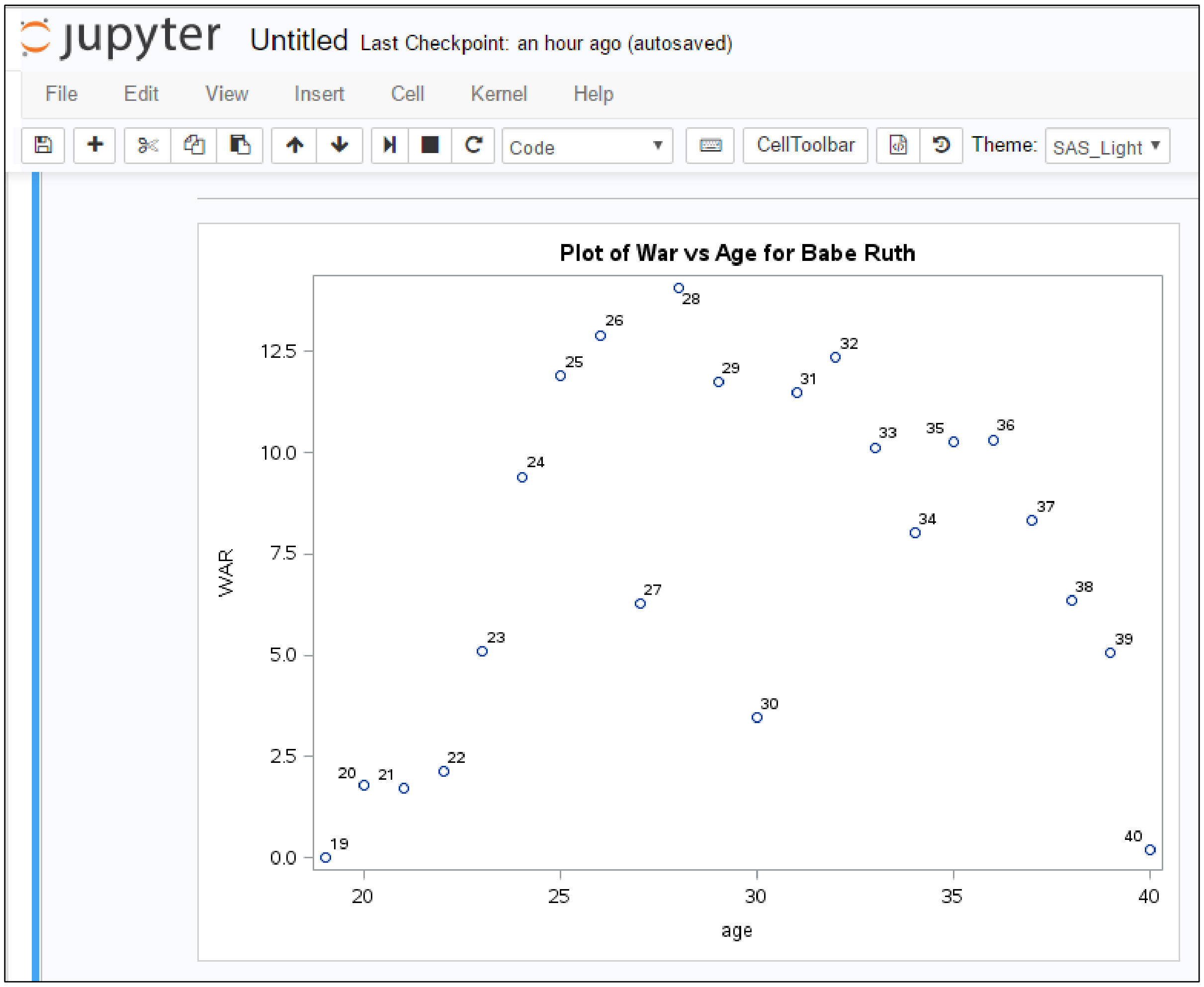
Distribution of WAR
With 95% Confidence Interval for Mean
Percent
25
20
15
10
5
0
Normal
Kernel
95% Confidence

Display 12. More Output from Executing SAS Code in Jupyter Notebook

Output 3 below is an additional example to show the similarities between SAS Studio and Jupyter Notebook output. It shows that Jupyter produces a graph that is almost identical to the SAS Studio graph. The graph in Output 3 was created by running the following code in Jupyter Notebook. Notice that the only difference in this code from what was run earlier in SAS Studio is the player ID and title.

```
proc sgplot data = baseball;
    scatter x = age y = war/datalabel=age;
    where player_id = "ruthba01";
    title "Plot of WAR vs Age for Babe Ruth";
run;
```

The scatter plot shows that the peak age in terms of WAR for Babe Ruth was 28.



Output 3. Scatter Plot of Age versus WAR for Babe Ruth

The key point of this example is to illustrate that you can use the interface provided by an open-source project to enter and run SAS code. For instructors or students who prefer the coding and learning environment provided by Jupyter Notebook, this is significant.

SAS E-LEARNING

You might find yourself asking, “Where should I start if I want to gain SAS skills?” There are two courses available on the SAS website that help you develop a groundwork: SAS Programming 1: Essentials and Statistics 1: Introduction to ANOVA, Regression, and Logistic Regression. The SAS Programming 1 course focuses on navigating the SAS environment, reading various types of data into SAS data sets, creating SAS variables and subsetting data, and producing and enhancing listing and summary reports. The Statistics 1 course helps build your knowledge of statistical procedures such as t tests, ANOVA, linear regression, and logistic regression. These resources give you a good foundation as you begin your journey with SAS programming.

SAS CERTIFICATION

If you want an official way to let the world know that you have worked hard to gain SAS skills, consider earning a SAS Certification. Although there is a fee for the certification exams, they can be a good investment in your career. SAS offers a number of certifications, but the first, most essential one is the SAS Certified Base Programmer for SAS 9. The free, e-Learning courses mentioned previously will help you prepare for this certification exam. You might also consider attending the SAS Programming 2: Data

Manipulation Techniques e-Learning course to complete your preparation. The combination of having hands-on experience programming in SAS and a SAS certification on your resume will give you the credibility and competitive advantage needed in today's demanding job market.

SAS SUPPORT COMMUNITIES

Anyone learning SAS will find great value in the SAS Support Communities site, and the SAS Analytics U community in particular. You can pose questions of all types, and get responses from knowledgeable SAS users. Typically, questions are answered very promptly. If programming tasks can be accomplished multiple ways, you will often get a response that includes several alternatives. Refer to "How to get fast, helpful answers" for guidance about successfully using the SAS Support Communities.

Note: Before posting a question, you should search the Communities to see if it has already been asked, both for the benefit of yourself and others.

CONCLUSION

This paper has discussed the motive for developing SAS skills, provided an overview of the tools available, and described some additional resources to supplement the learning experience. Being a successful student in any discipline involves focused time spent studying. If your goal is to become skilled in SAS and thereby increase your ability to analyze big data, you have a low-cost route to your destination with SAS University Edition, SAS e-Learning courses, and SAS Support Communities.

REFERENCES

Renzulli, K.A., Weisser, C., and Leonhardt, M. 2016. "The 21 Most Valuable Career Skills Now." *Money*, May 16, 2016. Available at time.com/money/4328180/most-valuable-career-skills.

SAS Institute Inc. 2016. *SAS University Edition: Help Center*. Cary, NC: SAS Institute Inc. Available at support.sas.com/software/products/university-edition/faq/main.htm.

Foley, Malachy J. 2008 "MERGING vs. JOINING: Comparing the DATA Step with SQL." *Proceedings of the Thirtieth Annual SAS Users Group International Conference*, Cary, NC: SAS Institute Inc. Available at www2.sas.com/proceedings/sugi30/249-30.pdf.

Droogendyk, Harry and Dosani, Faisal. 2008. "Joining Data: Data Step Merge or SQL?" *Proceedings of the SAS Global Forum*, Cary, NC: SAS Institute Inc. Available at www2.sas.com/proceedings/forum2008/178-2008.pdf.

SAS Institute Inc. 2016. *SAS® Studio 3.5: Task Reference Guide*. Cary, NC: SAS Institute Inc. Available at support.sas.com/documentation/cdl/en/webeditorref/68839/HTML/default/viewer.htm#titlepage.htm

SAS Institute Inc. 2013. *SAS® 9.4 Macro Language: Reference, Fifth Edition*. Cary, NC: SAS Institute Inc. Available at support.sas.com/documentation/cdl/en/mcrolref/67912/PDF/default/mcrolref.pdf.

FanGraphs. 2016. "What is WAR?" Available at www.fangraphs.com/library/misc/war.

wikiHow. 2015. "How to Calculate Baseball WAR." Available at www.wikihow.com/Calculate-Baseball-WAR.

Baseball-Reference. 2016. "WAR daily file (war_daily_bat.txt)." Available at www.baseball-reference.com/data.

Reese, George. "Scatter Plots" C&I 330MS, University of Illinois. 1997. Available at mste.illinois.edu/courses/ci330ms/youtsey/scatterinfo.html.

SAS Institute Inc. 2016. "SGPLOT Procedure" in *SAS® 9.4 ODS Graphics: Procedures Guide, Sixth Edition*. Cary, NC: SAS Institute Inc. Available at support.sas.com/documentation/cdl/en/grstatproc/69716/HTML/default/viewer.htm#p1t32i8511t1gfn17sw07yxtazad.htm.

Jupyter Notebook. 2016. "The Jupyter Notebook." Available at jupyter.org.

Brown, Beverly. “How to get fast, helpful answers.” SAS Institute Inc. 2015. Available at communities.sas.com/t5/Getting-Started/How-to-get-fast-helpful-answers/ta-p/226133.

RESOURCES

SAS University Edition:

- Download software: www.sas.com/en_us/software/university-edition/download-software.html
- FAQs: supportexp.unx.sas.com/software/products/university-edition/faq/main.htm
- Helpful video: www.youtube.com/watch?v=x5KwQp3Dh7g

SAS e-Learning courses: support.sas.com/edu/elearning.html?ctry=us

SAS Studio tutorials: support.sas.com/training/tutorial/studio/

SAS Certification: support.sas.com/certify/

Jupyter Notebook:

- jupyter.org/
- blogs.sas.com/content/sasdummy/2016/04/24/how-to-run-sas-programs-in-jupyter-notebook/
- github.com/sassoftware/sas_kernel

SAS Customer Support site: support.sas.com

Baseball Reference WAR data: www.baseball-reference.com/data/

CONTACT INFORMATION

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

Randy Mullis

Allison Mahaffey

Parth Patel

SAS Institute Inc.

SAS Campus Drive

Cary, NC 28513

Email: support@sas.com

Web: support.sas.com