PharmaSUG 2026 - Paper AP-274

Experience of an R Programmer Incorporating R in a SAS Studio Flow

Shelby Taylor, SAS Institute

ABSTRACT

Integrating open-source tools into enterprise analytics workflows allows you to increase flexibility and reproducibility in your statistical programming. In this paper, you will learn how to incorporate native R code into a low-code pipeline environment by using the R Runner custom step within SAS Studio on SAS Viya. You also will preview how the new R procedure will provide additional options for integrating R code into SAS. The paper walks through examples of how to fit regression models using base R, transform data with the tidyverse set of packages, and produce visualizations with ggplot2. All analyses run within SAS Studio and remain interoperable with SAS data sets.

INTRODUCTION

When discussing programming languages, people often position R and SAS as competitors. If you primarily program in R and are new to SAS—or vice versa—you might be surprised by how many ways you can combine the strengths of both languages.

With SAS Studio, you can integrate Python and R code within SAS workflows. Using the R Runner custom step, you can write R code directly or upload an entire R script to execute within a stand-alone tab or a flow. In addition, SAS is expanding these capabilities with a new R Program step, slated for release in Q2, which provides a more streamlined and fully integrated way to run R code within flows.

In this paper, you will learn how to use R Runner while also gaining an understanding of the upcoming R Program step and how these tools can support anyone building analytic workflows in SAS Studio.

SAS STUDIO FLOWS

If you are asking yourself, “What is a ‘flow’?”, SAS Studio flows offer a low-code option for analyzing data with SAS Viya. A flow is a sequence of operations on data. Data and operations are represented in SAS Studio by steps that you can access from the Steps section of the navigation pane. Each step in a flow is represented by a node on the flow canvas.

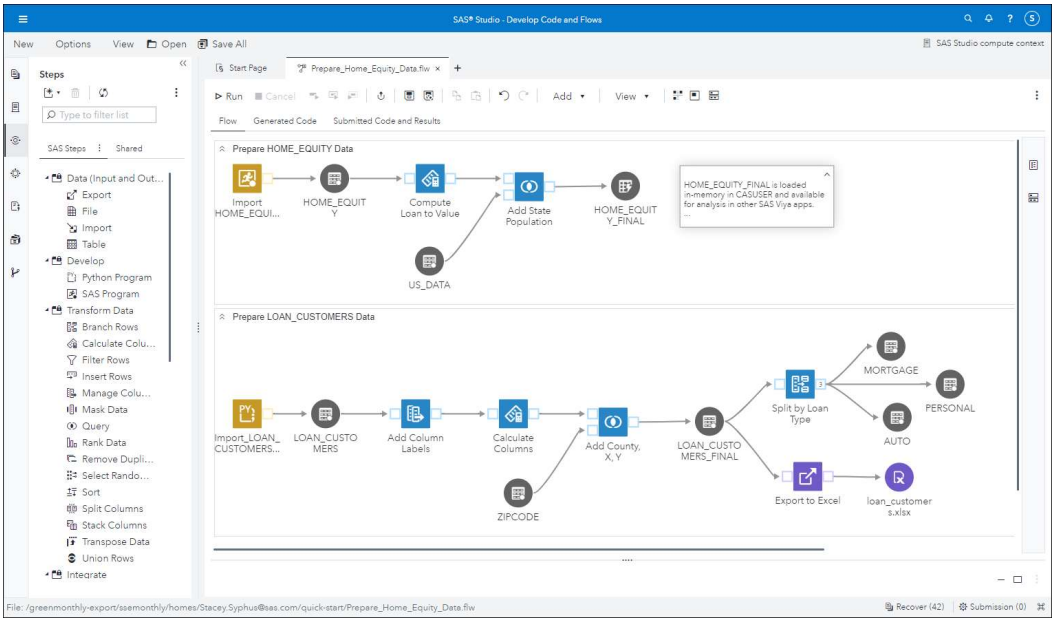

Figure 1: SAS Studio Flow Example

CUSTOM STEPS

SAS provides several predefined steps that can be used within a flow or in a stand-alone tab in SAS Studio. Developers can also build custom steps to perform specific tasks. Custom steps include a point-and-click form allowing a user to enter selections and customize options, and corresponding code behind the scenes that is modified based on the user selections. R Runner is a custom step that allows users to run R scripts and write R code to analyze data. While the R Runner custom step can be used both within a flow and in a stand-alone tab in SAS Studio, this paper will focus on using the step within a flow.

SET UP

You can access R Runner through the SAS Studio Custom Steps GitHub repository which is available to all users.

Before using R Runner, make sure your environment meets the necessary requirements. Work with your SAS administrator to confirm the following prerequisites:

- The SAS Viya version must be 2023.08 or later.
- SAS Viya needs access to an active Python and R environment.
- The rpy2 Python package must be installed and configured.
- A path to R is available through the R_HOME environment variable. The option to run R within SAS Studio compute sessions is enabled with the -RLANG system option.
- **Recommended:** Administrators could make use of the SAS Configurator for Open Source (also commonly known as sas-pyconfig) to install and configure Python and R access from SAS Viya.

CREATING A FLOW

To create a flow, you navigate to **Develop Code and Flows** from the Applications menu in SAS Viya.

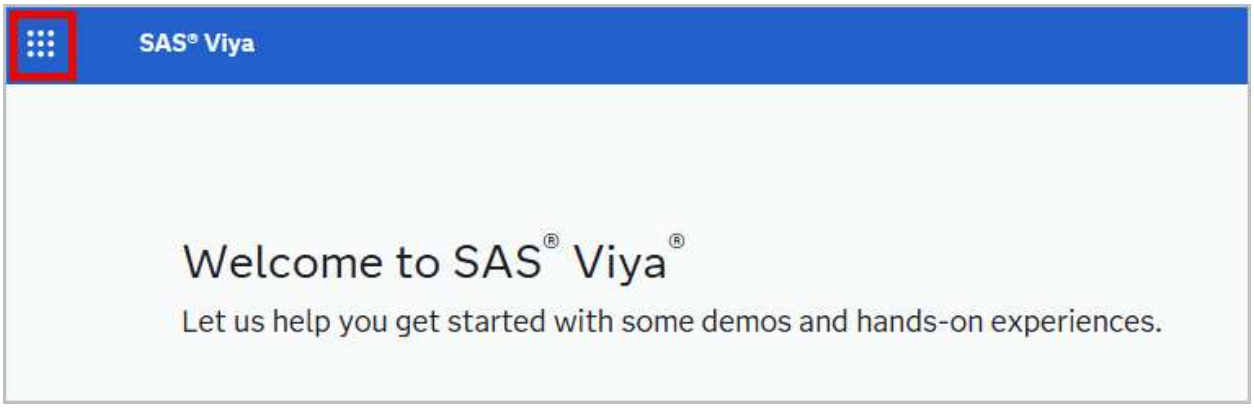

Figure 2: Applications Menu

Figure 3: Develop Code and Flows

Next you click New > Flow.

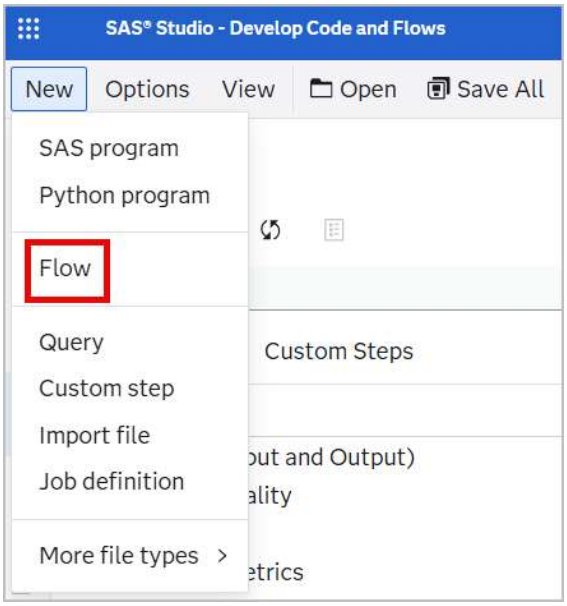

Figure 4: New Flow Creation

From here you can access steps from the **Steps** tab on the lefthand pane. Because R Runner is a custom step, you navigate to the **Custom Steps** tab and search for R Runner. From here, you can either click and drag the step into a flow or open the step in stand-alone mode by right-clicking and selecting **Open**.

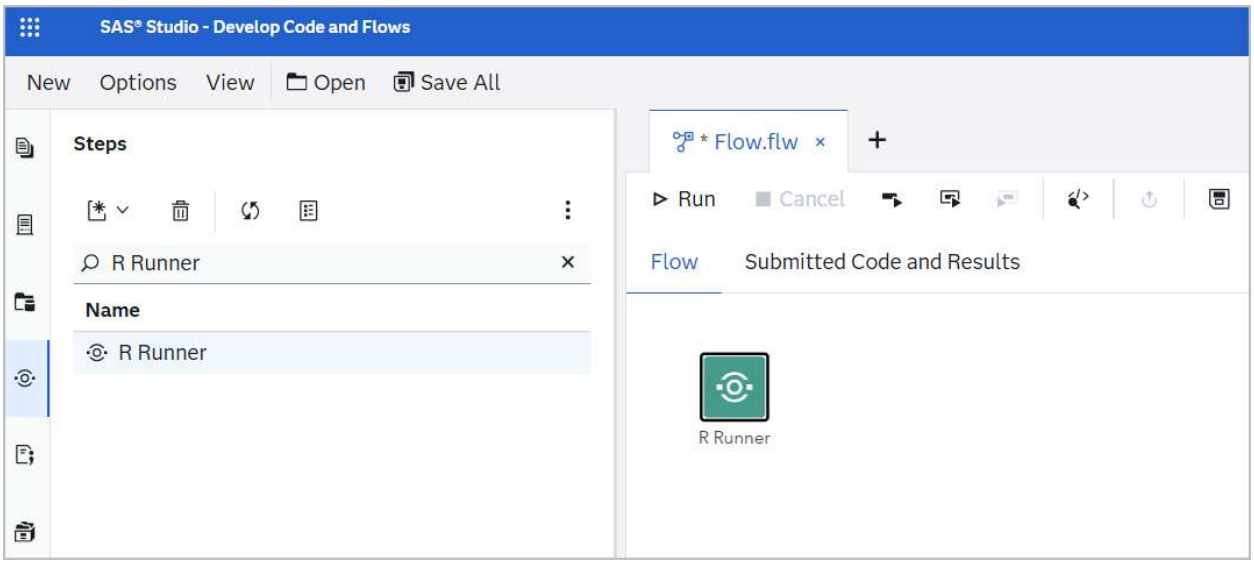

Figure 5: Adding R Runner to a Flow

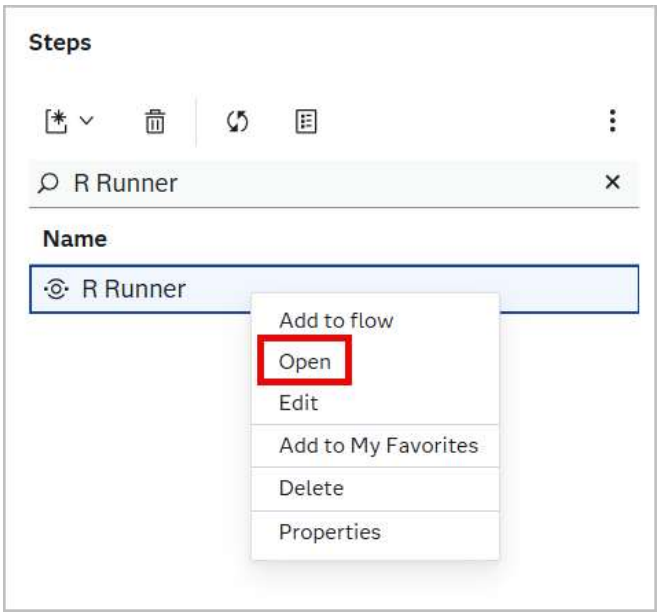

Figure 6: Open R Runner in Stand-Alone Mode

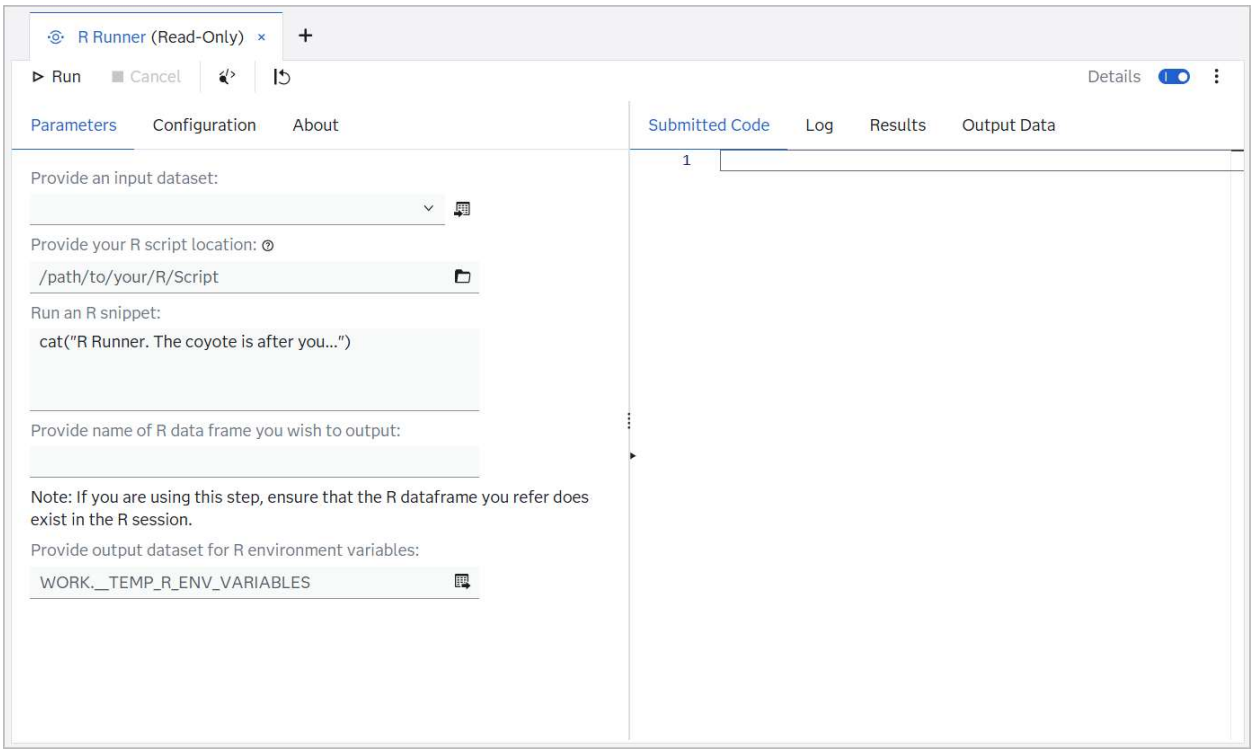

Figure 7: R Runner Open in Stand-Alone Mode

USING R RUNNER

Now that R Runner is ready to use, you might start your exploration the same way many programmers do - by entering the classic line **"Hello World!"** in the text box to confirm that everything is working.

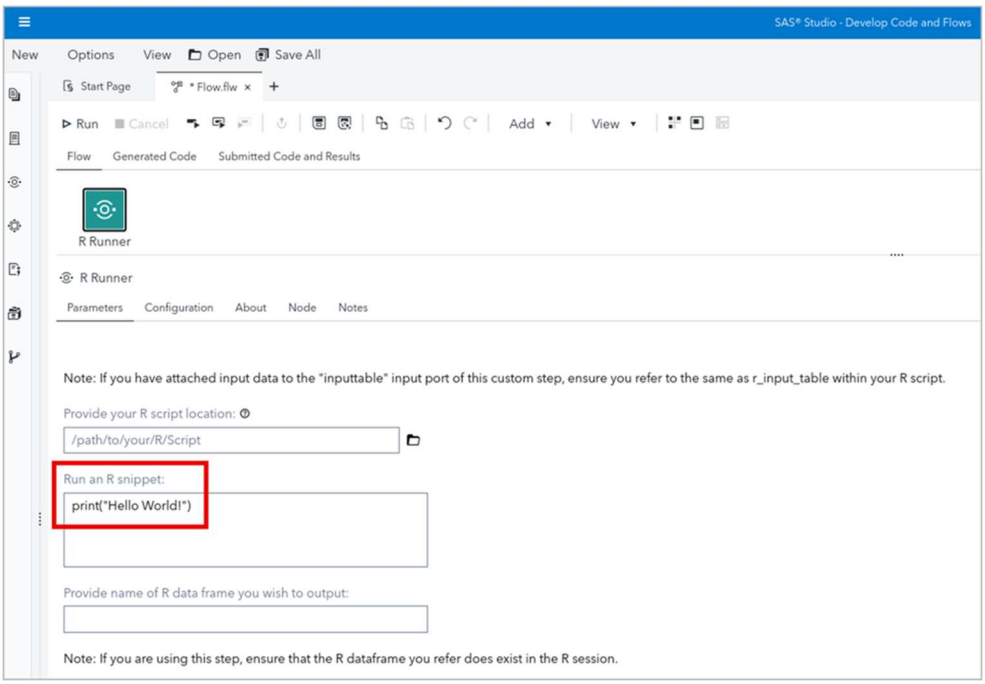

Figure 8: Hello World! Code

Navigating to the **Submitted Code and Results** log, you find the output of your R Code.

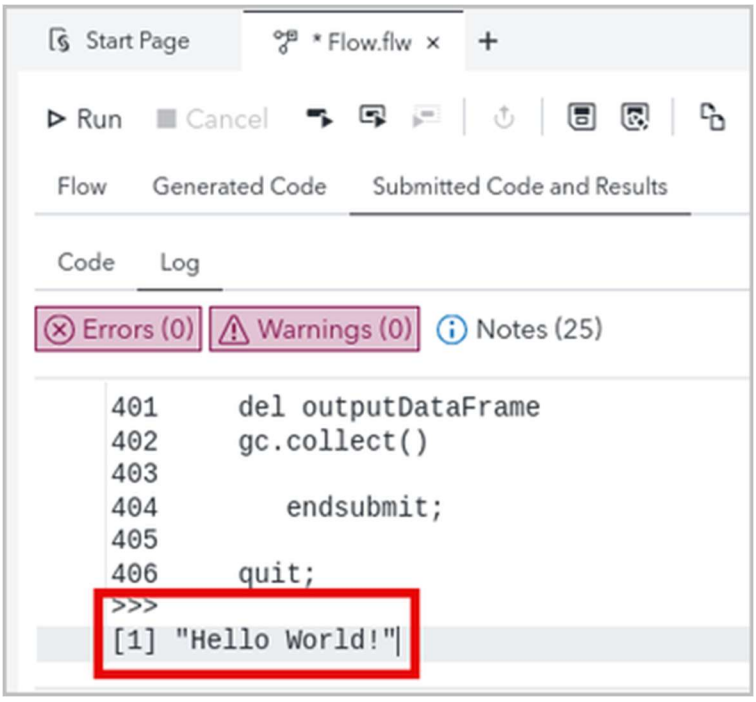

Figure 9: Hello World! Results

Next, you can submit a regression model to analyze the SASHELP.CARS table and examine the relationship between Horsepower and Highway MPG by following these steps:

1. Right-click on the R Runner step and select **Add input port.**

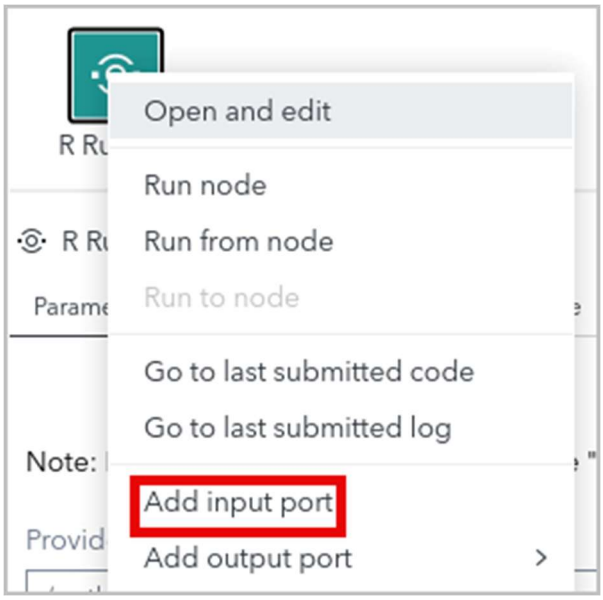

Figure 10: Add Input Port

2. From the Libraries pane, select the SASHELP.CARS table and drag it to the flow and connect it to the R Runner input port.

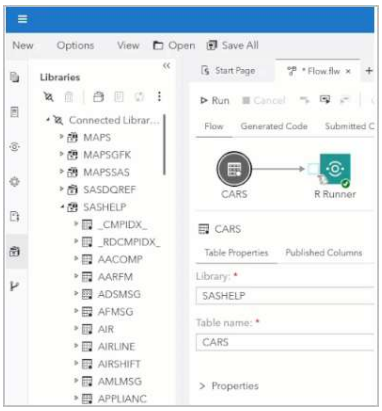

Figure 11: Connecting SASHELP.CARS to R Runner

3. Add the model code to the textbox. Notice instead of supplying CARS for the table name, you specify `r_input_table`.

```
model <- lm(MPG_Highway ~ Horsepower, data=r_input_table)
print(summary(model))
```

Program 1: Linear Model

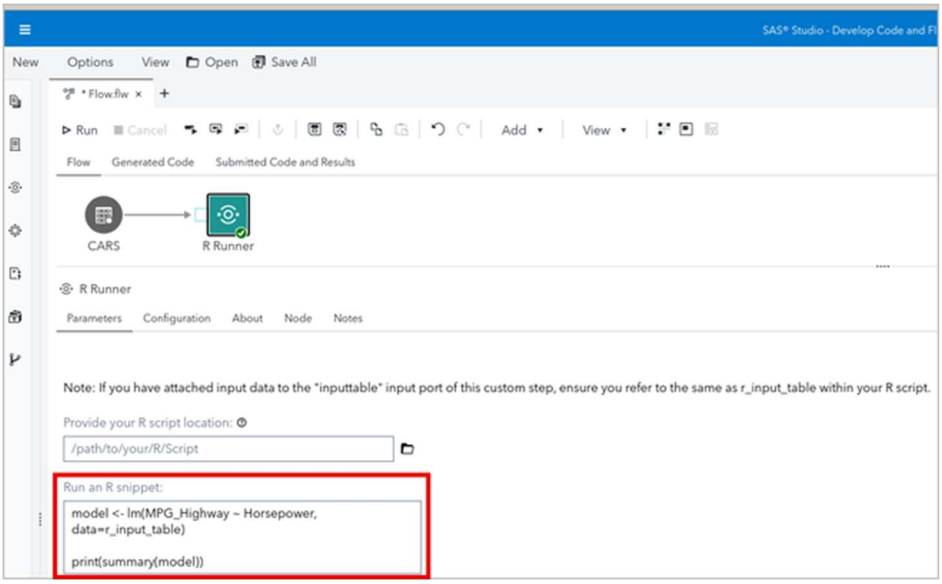

Figure 12: Run an R Snippet

4. Run the flow and navigate to the **Submitted Code and Results** log. The program returns a summary of the regression model.

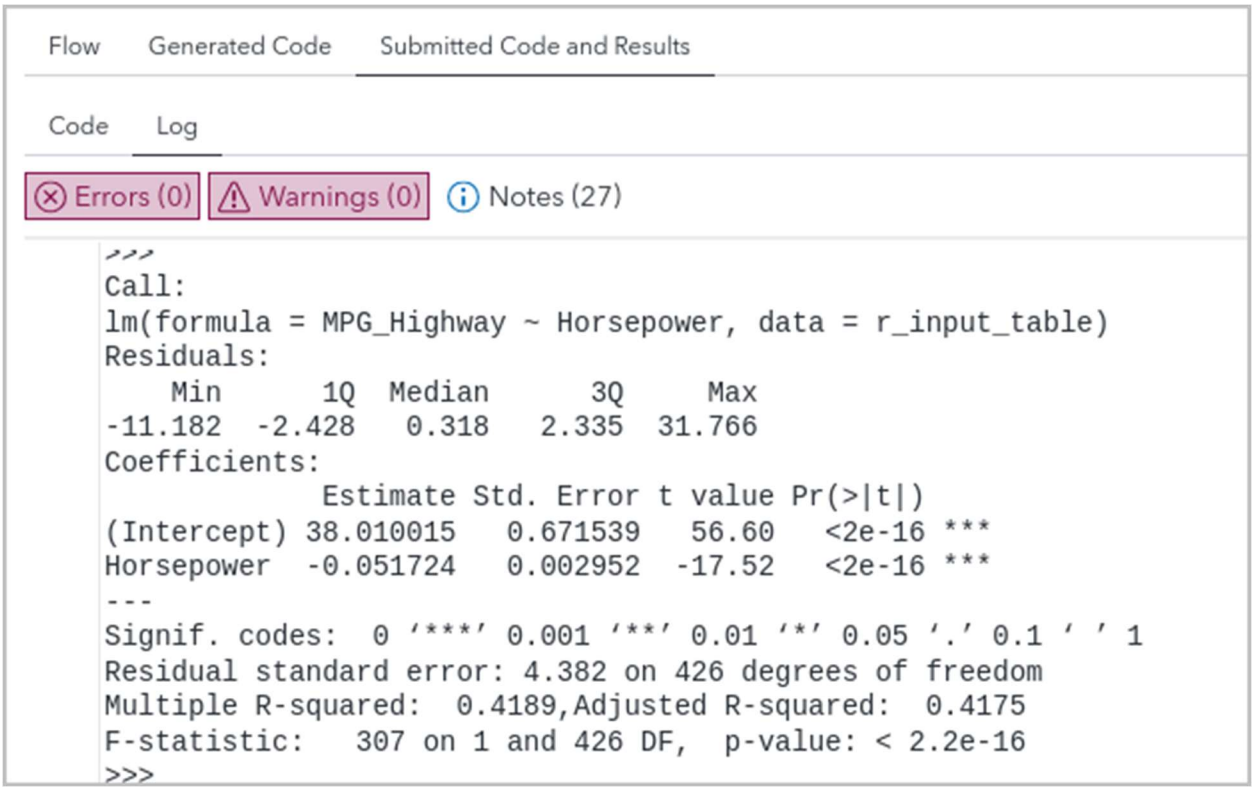

Figure 13: Model Results

In addition to printing R output, R Runner can modify SAS tables with R code within a flow. For example, let's say you want to filter the SASHELP.CARS table by Make, only keeping Audis. The *tidyverse* set of packages in R includes convenient data transformation functions.

tidyverse was installed with the SAS Open Source Configurator so all you need to do is build your flow and include library(tidyverse) in your R code to load in the *tidyverse* packages. To create a table with only Audi vehicles included, you follow these steps:

1. Follow steps 1 and 2 of the previous example to connect SASHELP.CARS to R Runner.
2. Right-click R Runner and add an output port, selecting **outputtable.**

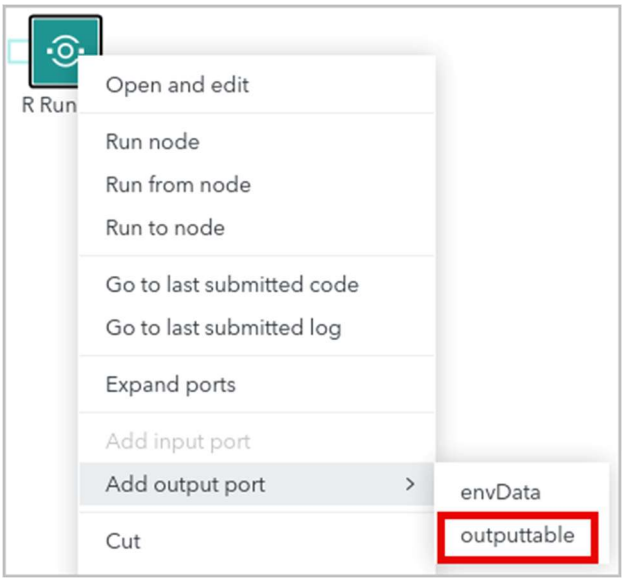

Figure 14: Add Output Port

3. Right-click the output port and add a table, providing the library WORK and table name AUDI.

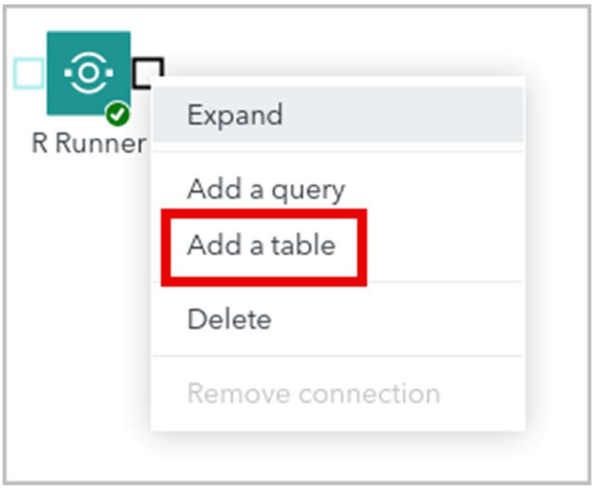

Figure 15: Add a Table

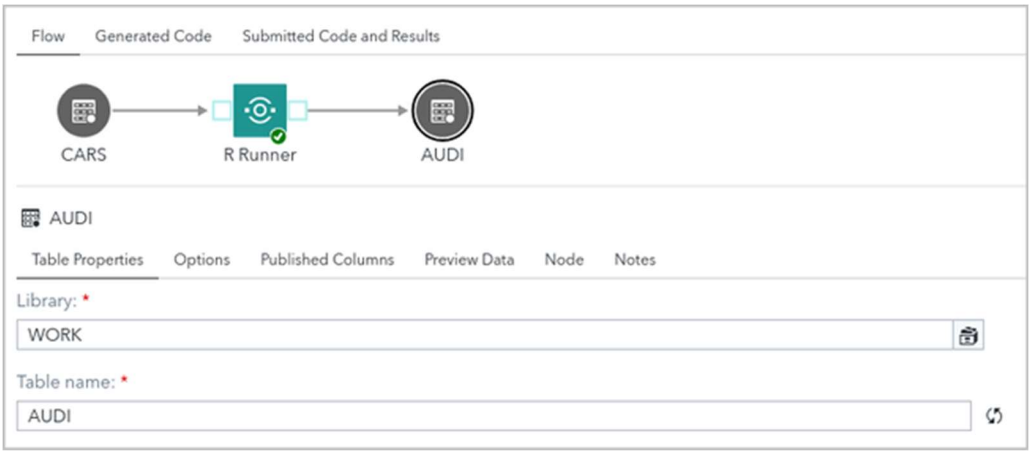

Figure 16: Flow

4. Click R Runner again and add code to the textbox.

```
library(tidyverse)
AUDI <- r_input_table %>% filter(Make == "Audi")
```

Program 2: Filter Data

5. Provide a table name of AUDI to the **Provide name of R data frame…** textbox. This is used by R Runner to write to the defined output table.

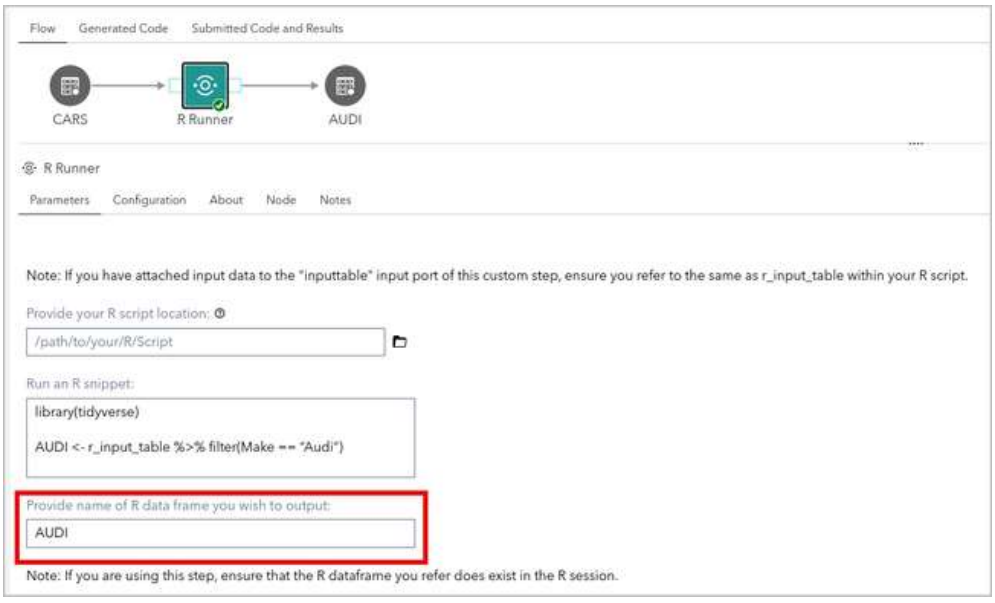

Figure 17: Table Name

6. Run the flow and navigate to the work library to open WORK.AUDI.

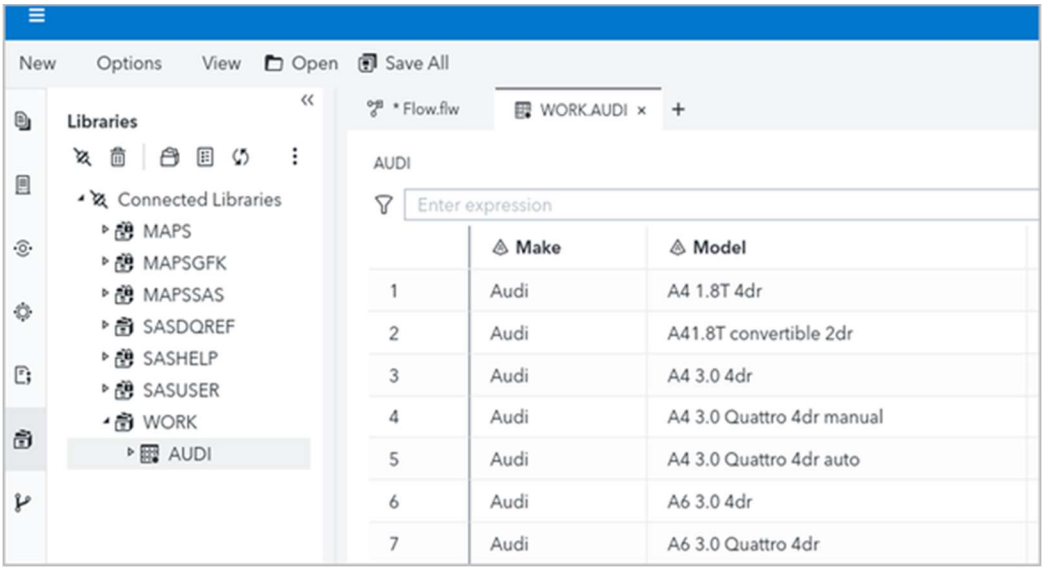

Figure 18: View New Table

SUBMITTING AN R SCRIPT

Next, you put the same code in an R script and provided the location of the script to R Runner which gives the same results. This method is especially helpful to run longer programs and allows you to use the editing capabilities provided in an IDE such as R Studio to build the script, then execute it within a flow.

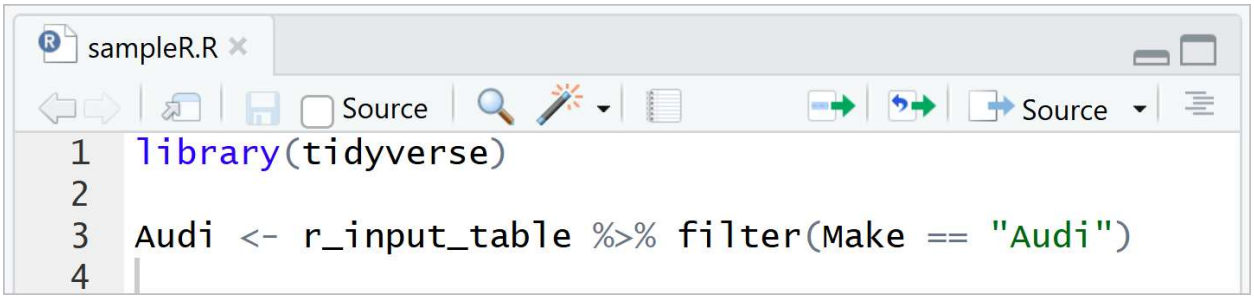

Figure 19: R Script

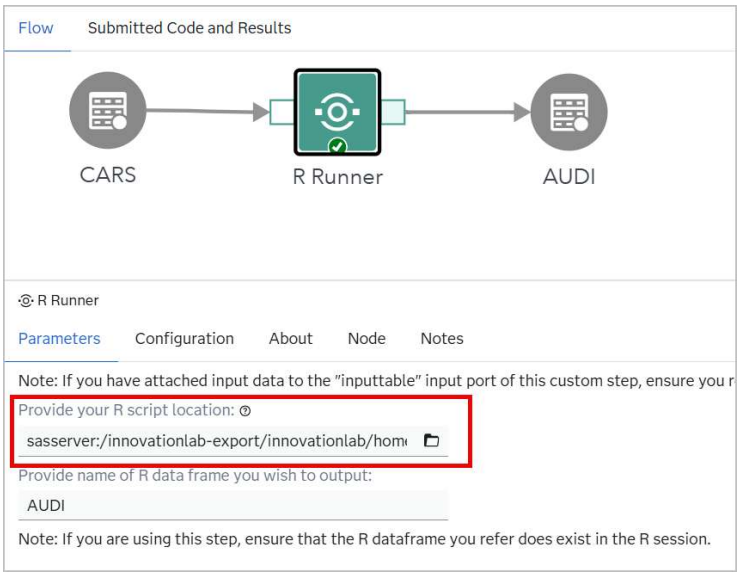

Figure 20: Path to R Script

DATA VISUALIZATION

You can even take advantage of R's graphing capabilities within a flow. Below is an example creating a histogram using the ggplot2 package in R. This feature is limited to creating PDFs currently, but with the introduction of the R Program step, users will be able to view graphics in the Results tab.

You can create a histogram of the Highway MPG data from SASHELP.CARS following these steps:

1. Connect the SASHELP.CARS dataset to R Runner following steps 1-2 from the previous examples.
2. Add code to create and save the plot in the textbox.

```
library(tidyverse)
AUDI <- r_input_table %>% filter(Make == "Audi")
```

Program 3: Create and Save GGPLOT

3. Locate the PDF in your file directory.

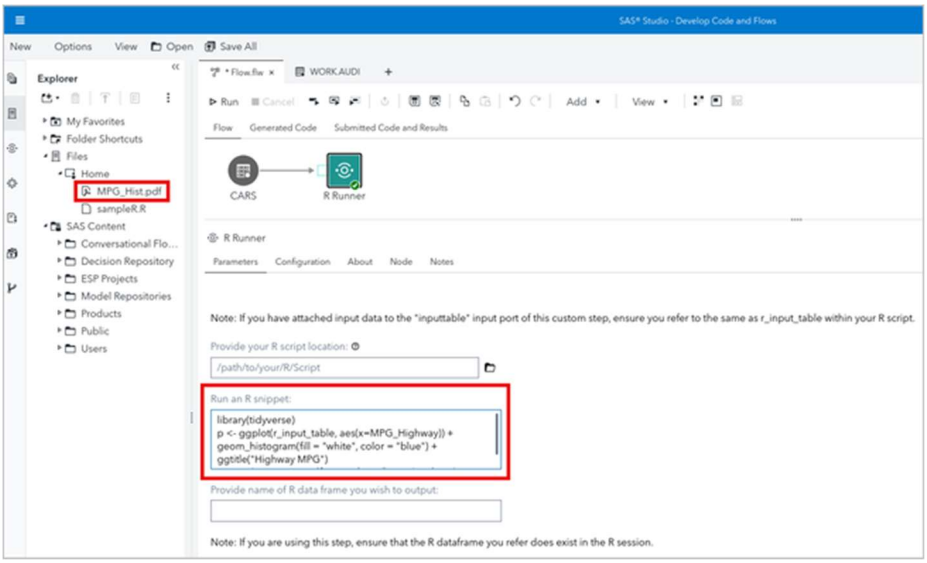

Figure 21: PDF Location and Creation Code

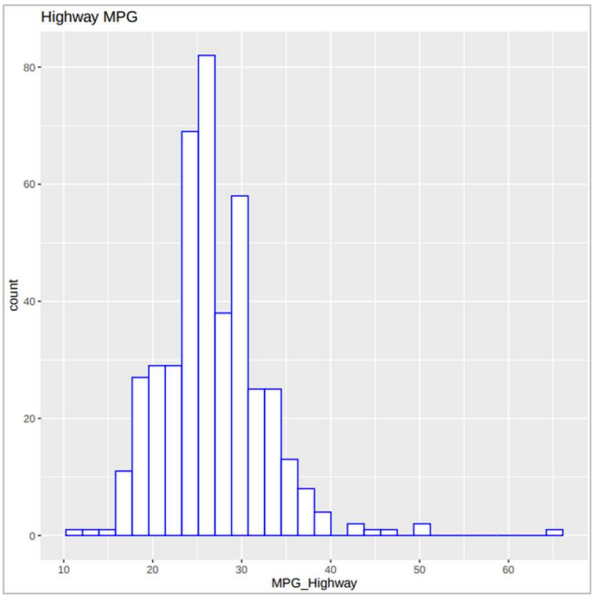

Figure 22: MPG Histogram

INSTALLING R PACKAGES

When configuring R with SAS Viya using the SAS Configurator for Open Source, certain R packages are suggested as part of the installation manifest. If there are specific R packages you know you will need to use as part of your R script, please let your SAS administrator know so they can include those packages in the configuration. As of now, users cannot install R packages directly using R Runner.

CURRENT DEVELOPMENTS

An exciting new R procedure (PROC R) became available in the 2026.03 LTS release of SAS Viya. With the recent release of PROC R, SAS users now have even more flexibility in how they integrate R within SAS code. An upcoming R Program step, slated for release in Q2, will allow users to run R code directly or connect to and execute external R scripts within a flow. This step offers several advantages, including the ability to output tables and graphics to the Results tab rather than the Log.

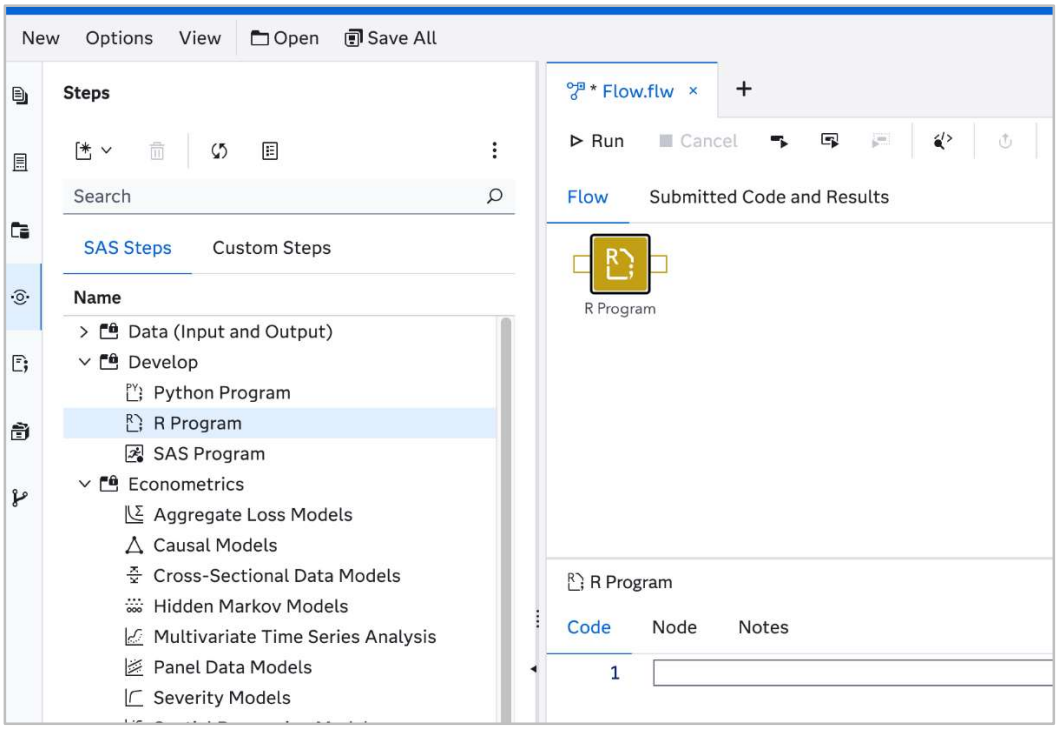

Figure 23: Preview of the R Program Step

Users can also create their own custom steps to be used within a flow. For example, here is a custom step called Rplot_step that allows users to create plots with R, choosing their graph of choice and corresponding variables

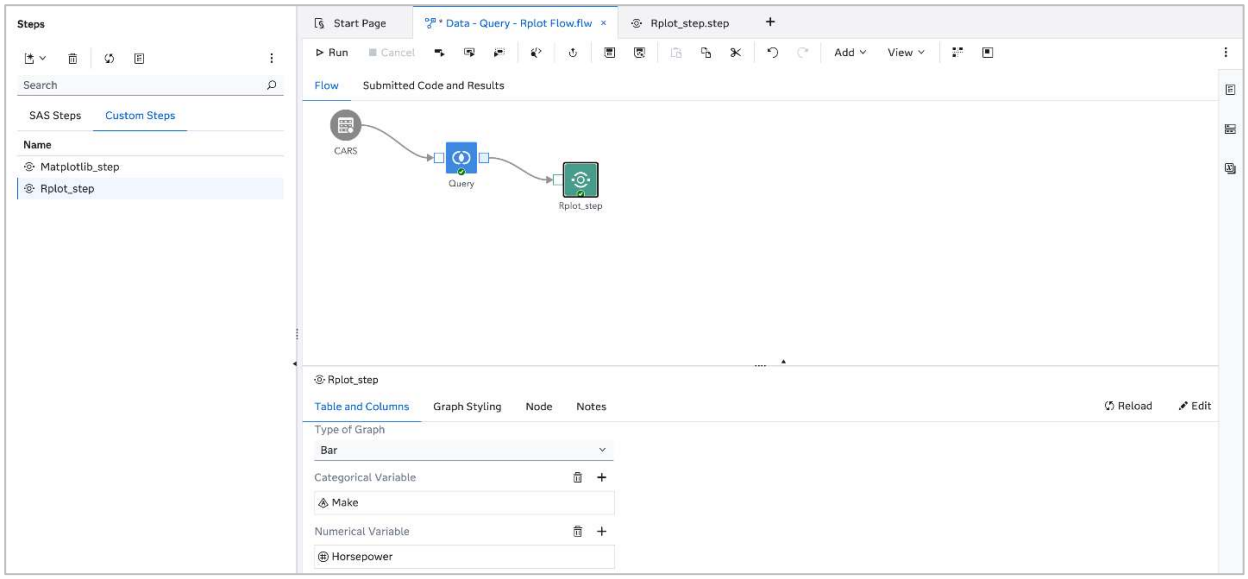

Figure 24: Example Custom Step Specifications

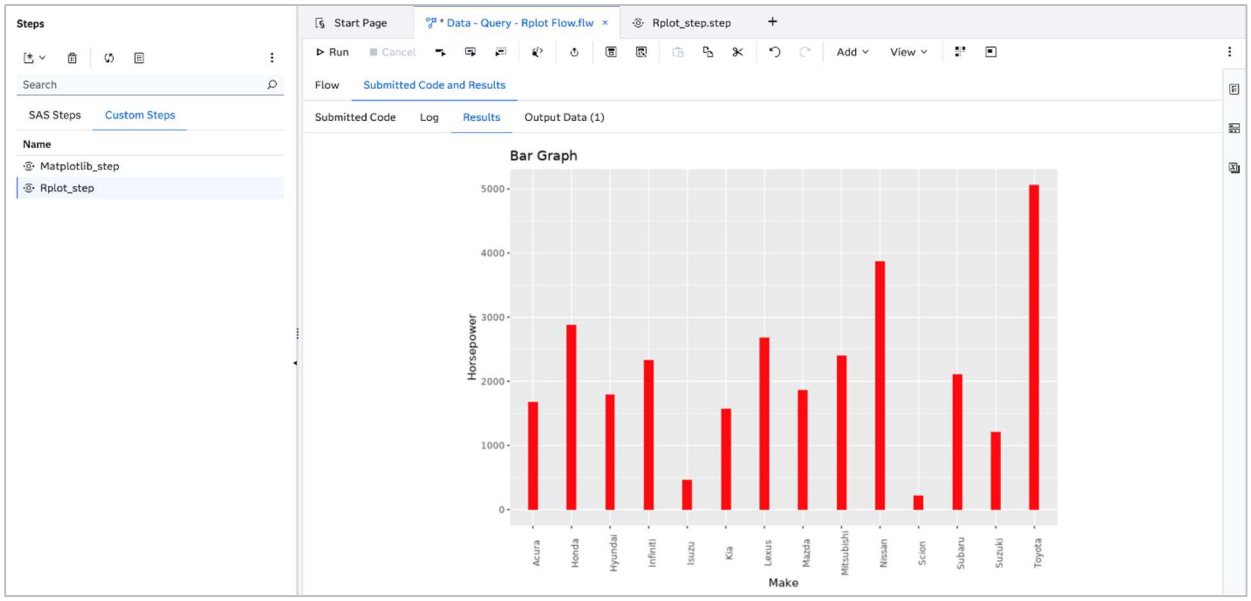

Figure 25: Custom Step Results

CONCLUSION

R can be effectively incorporated into SAS Studio Flows using the R Runner custom step, enabling users to combine the flexibility of R with the structure and scalability of SAS Viya. Through examples in modeling, data transformation, and visualization, you saw how R code can seamlessly operate within a SAS-driven workflow while maintaining interoperability with SAS data. As SAS continues to evolve with the introduction of PROC R and the upcoming R Program step, users can expect an even more streamlined and integrated experience for executing R within flows. Together, these tools expand the possibilities for building flexible, reproducible, and collaborative analytics pipelines.

REFERENCES

SAS Institute. (n.d.). *R*. SAS Developer Portal. https://developer.sas.com/open-source/r

SAS Institute Inc. (n.d.). *Build flows with SAS Studio* [Video]. SAS Video Portal. https://video.sas.com/detail/video/6325462242112/build-flows-with-sas-studio

SAS Institute Inc. (2023). *SAS Studio: Working with flows*. https://go.documentation.sas.com/doc/en/webeditorcdc/v_064/webeditorflows/titlepage.htm

Sankaran, S. and Haque, S. (2023, August 27). *R Runner: Get from SAS to R through a Python tunnel*. SAS Communities. https://communities.sas.com/t5/SAS-Communities-Library/R-Runner-get-from-SAS-to-R-through-a-Python-tunnel/ta-p/890927

SAS Institute Inc. (n.d.). *R Runner* (README). GitHub. https://github.com/sassoftware/sas-studio-custom-steps/blob/main/R%20Runner/README.md

ACKNOWLEDGEMENTS

I'd like to thank Stacey Syphus and Elijah Reid for their contributions to this paper.

CONTACT INFORMATION

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

Shelby Lynne Taylor
SAS Institute
shelby.taylor@sas.com