

Low-Code Approach to Clinical Application Development Using SAS Custom Step

David Olaleye, SAS

ABSTRACT

The SAS Custom Step feature offers a quick low-code application development platform that programmers can use to create code snippets and a user interface on top of the code snippets. It can be used to create dynamic user interfaces (UI) for end users with little or no programming experience to gain access to data assets at their site and perform tasks such as data exploration, analysis, and visualization. Custom steps come with cascading prompts and prompt hierarchies that enable the creation of data dependencies between control objects, thus enhancing the user experience as they query data and interact with the application. In this paper, I will show how to create a stand-alone SAS custom step for a demographic table and subgroup summary report and present another custom step for performing a propensity score match analysis. Finally, the two custom steps are added to a SAS® Studio Flow to show how custom steps can facilitate and enhance a reporting and analysis process workflow.

INTRODUCTION

Low-code and no-code tools are becoming increasingly popular for automating business processes and workflows. They offer developers and users a quick and easy way to build custom applications. Custom steps are a feature in SAS Studio that enables users with minimal coding skills to create code snippets and user interfaces on top of the code snippets to automate complex and repetitive tasks. These snippets, which can contain SAS or python code, are known as custom steps. To use custom steps, users need either a SAS Studio Analyst or a SAS Studio Engineer license. The options available to users depend on the version of the SAS Studio license available at their site. For more information, visit <https://documentation.sas.com/doc/en/sasstudiocdc/default/webeditorcdc/webeditorsteps/n1rff6jqdi2dt4n1ur1avntzf2q0.htm>.

OVERVIEW OF SAS CUSTOM STEPS

WHAT IS A CUSTOM STEP

A custom step enables you to create a user interface for users with minimal or no code experience in SAS programming language to complete a specific task. There are three benefits to using custom step functionality for your next code development project.

- No Code Low Code (NCLC) – The Step Designer UI enables the step creator to build a point-and-click user interface on top of their code snippets or create a stand-alone application that enables users to access information from different data sources at their site.
- Code Reusability and Shareability – Custom step is tightly integrated with SAS Studio, thereby allowing step creators to save their work to any file system on a SAS server or in SAS Content, which makes the steps available to SAS Studio users.
- Git Repository and Team Collaboration – Step creators can create, publish, and share their work on GitHub with the community. GitHub provides opportunity to collaborate, share knowledge, experience, and best practices with other step authors. There is also a custom step repository (<https://github.com/sassoftware/sas-studio-custom-steps>) on GitHub featuring published custom steps from simple to advanced data transformation steps, to statistical, data and text mining steps.

SAS CUSTOM STEP AS A LOW-CODE SOLUTION DEVELOPMENT TOOL

Custom steps are accessed via SAS Studio, which users can access through a web browser. The step creator can use the step **Designer** UI and other controls to create a simple application or a customized

user interface for end-users to perform a custom task such as run a report, query a database, or perform statistical data analyses.

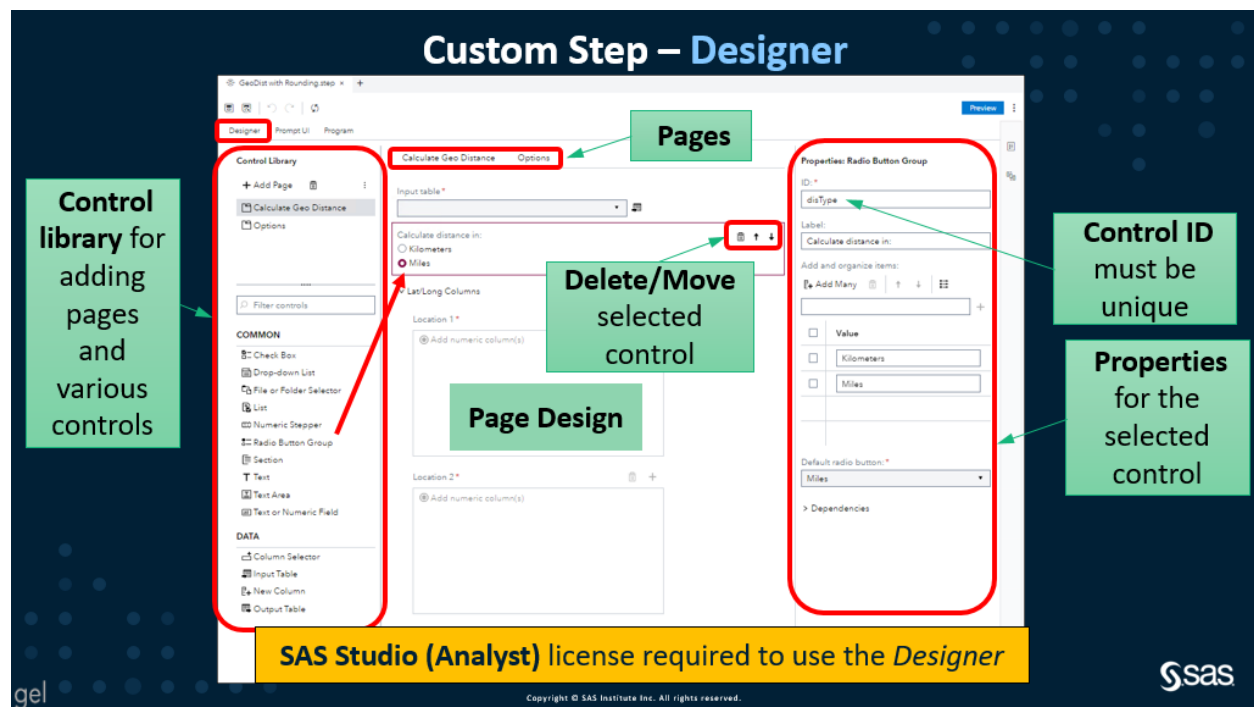
CREATING A CUSTOM STEP

There are two parts to creating or authoring a new custom step. The first one is defining the UI using the **Designer** interface. The second is writing the SAS or python code, which is done in the **Program** tab of the custom step definition. For every custom step, the UI syntax is an object in JSON. This code is generated by SAS Studio when you create a custom step using the **Designer** tab. Step creators with knowledge of JSON syntax can also create custom steps by adding JSON code to the **Prompt UI** tab. (I provide an example of how to do this in the paper.)

SAS CUSTOM STEP BUILDING BLOCKS

The building blocks for creating a new custom step consist of the **Designer** tab, the **Prompt UI** tab, and the **Program** tab. In this section, I give a brief description of what each tab does. In the example use case section, I provide a detailed walk-through on how to use these components to build and create a custom summary report and perform a statistical analytical task.

The **Designer** tab contains the visual and non-visual controls that the step creator can use to build the user interface for their code snippets. The step author uses the controls to capture input and display information to the user. As shown in Display 1, the controls library displays the UI elements that can be used to select and create pages, access tables in SAS libraries, files in directories, columns from a table, drop-down lists, radio button and check box objects for users to interact with the step at run-time. The **Designer** UI also enables the step creator to establish dependencies amongst the UI elements.



Display 1. SAS Custom Step Designer

The **Dependencies** feature enables the step author to specify the state of one control based on the values of other controls. For example, the selection of a check box control could result in the visibility of a drop-down list control. If the check box control is not selected, the drop-down list control remains hidden.

The **Prompt Hierarchies** feature is used to create a hierarchy of dynamic controls. The hierarchy feature indicates how the value of one dynamic control affects the available values in another or subsequent dynamic control. For example, if the user selects the matching variable for the propensity score model,

the corresponding values or levels of the selected matching variable are displayed in the drop-down list object below.

The **Prompt UI** tab contains the JSON code generated by SAS Studio when controls are added to the Designer UI. Step creators who are familiar with JSON code can also create a custom step by writing this JSON code themselves. To view the JSON code for each control, open the **Prompt UI** tab.

The **Program** tab defines the code section part of the step. In this tab, the step creator adds the SAS code for the step. After you add the custom step to a flow and the flow runs, the SAS code runs. When the flow runs, macro variables are generated for each control on the **Designer** tab. When writing the SAS code, use the macro variables to reference the prompts. The SAS macro variables represent values specified by the user in the UI at run-time.

RUNNING A SAS CUSTOM STEP

A custom step can be executed either in a stand-alone mode or when added as part of a SAS Studio flow.

RUNNING A CUSTOM STEP IN STAND-ALONE MODE

Running a custom step in stand-alone mode means that the user is presented with a new tab in SAS Studio that contains the UI for the custom step. The user fills in the required fields and clicks the run button. The results, either an output table or a report generated by the SAS code, are shown in the SAS Studio UI. An example of what the step results might look like is shown in Figure 1. Using stand-alone mode to execute a step gives the user the option to refresh the custom step definition, view the code and the generated log, and to save the custom step as a SAS program. After a custom step is executed, the results appear in the tab for the custom step. Graphical output appears on the **Results** tab, and output data appears on the **Output Data** tab. The user can open the same custom step multiple times in the SAS Studio workspace. When there are multiple instances of the custom step open at the same time, the user can provide different values for the controls and quickly compare the results.

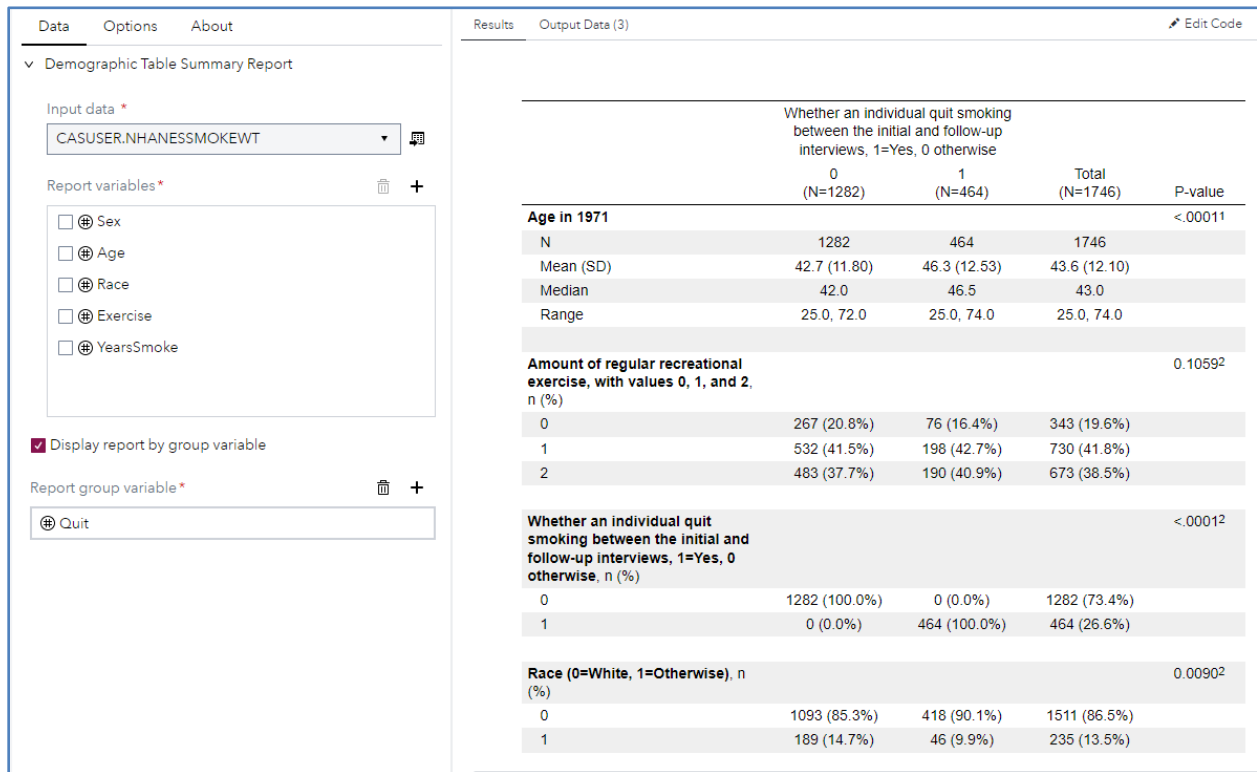


Figure 1. Results from a custom step in stand-alone mode

RUNNING A CUSTOM STEP IN SAS STUDIO FLOW

Running a custom step in flow mode enables the user to perform a sequence of operations on data using a graphical flow builder. SAS Studio provides an out-of-the-box library with steps that perform operations on data. Figure 2 shows an example of a custom step defined as part of a SAS Studio flow.

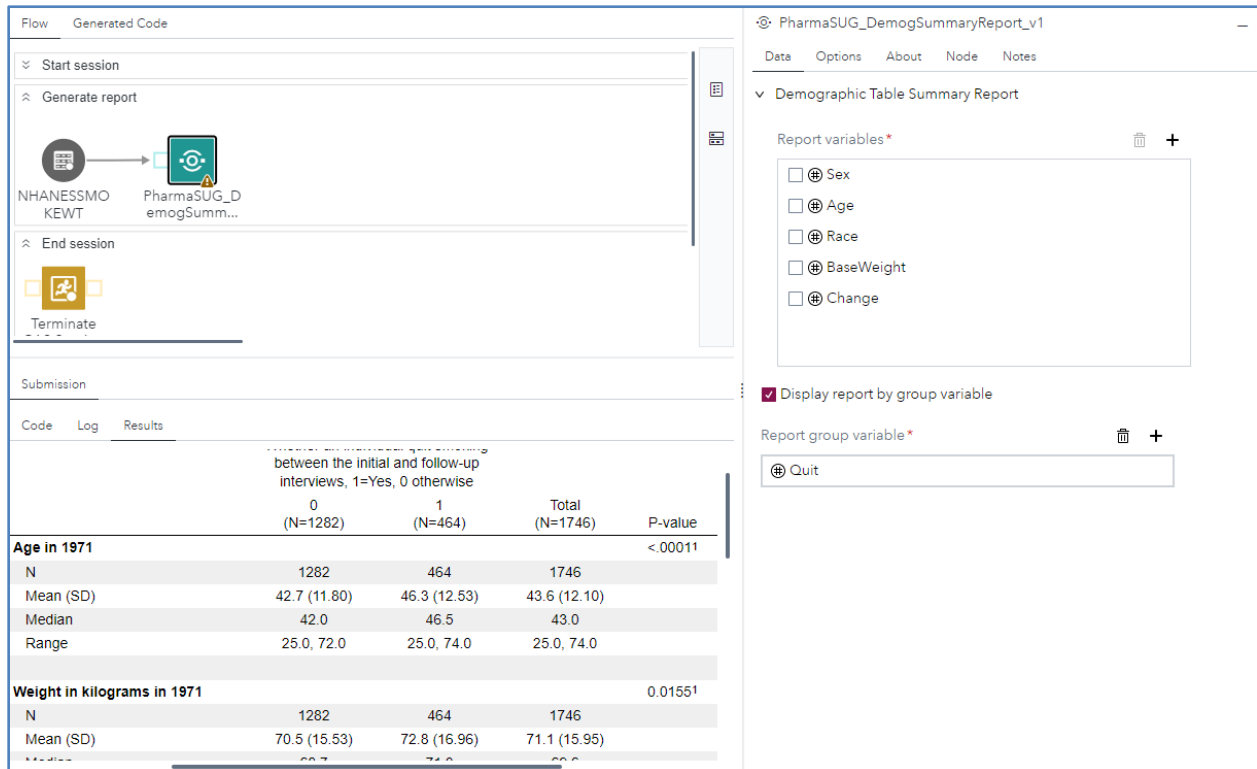


Figure 2. Results from a custom step as part of a SAS Studio flow

EXAMPLE USE CASES

I provide two example use cases to show how to create stand-alone SAS custom steps: the first one is for a demographic table and subgroup summary report and the second use case highlights a custom step for performing a propensity score match analysis.

USING A CUSTOM STEP TO CREATE A DEMOGRAPHIC TABLE AND SUBGROUP SUMMARY REPORT

This example uses a custom step to create the user interface that generates the demographic table and subgroup summary report. The report uses the %TABLEN SAS macro written by Jeff Myers and presented at PharmaSUG 2020. As stated by Jeff, the %TABLEN macro is a tool developed to compute distribution statistics for continuous variables, date variables, discrete variables, univariate survival time-to-event variables, and univariate logistic regression variables and combine them into a table for publication. To use this macro in its present form requires knowledge of how SAS macro language works. A user must also supply the required parameters for the %TABLEN macro. This example uses the

custom step feature to create a UI to make the macro easily accessible and usable for end-users with little or no SAS code or macro programming language background.

When a user runs this custom step, the result is a table or a report (or both) that shows the distribution information such as frequencies for categorical and binary variables, and means, medians, and ranges for continuous and date variables (see Figure 3). When a BY group variable is provided as a treatment variable, the custom step report can be used to compare the distributions across the treatment arm and to determine whether there was any imbalance in the sample populations being compared. Details about the %TABLEN macro are available here: <https://communities.sas.com/t5/SAS-Communities-Library/Demographic-Table-and-Subgroup-Summary-Macro-TABLEN/ta-p/634030>.

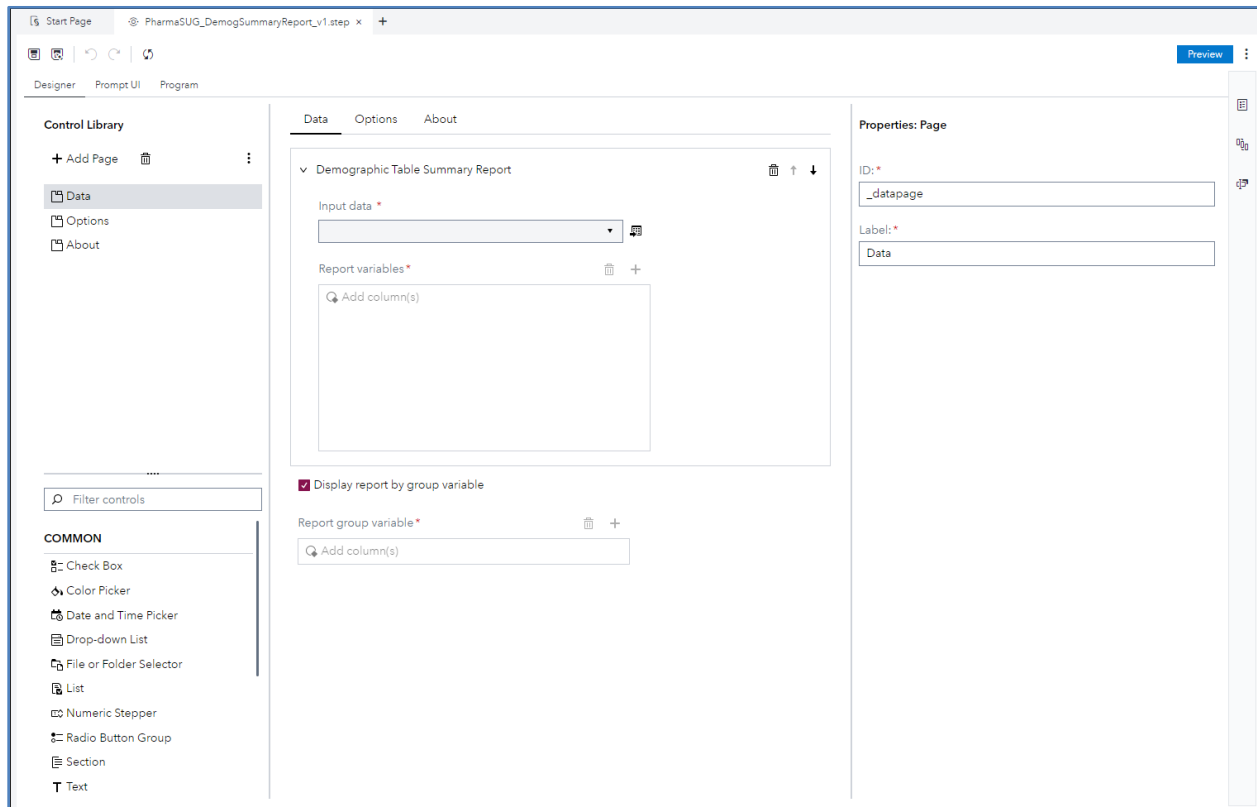


Figure 3. Custom Step Designer UI for Demographic Table and Subgroup Summary Report


Overview of Demographic Table and Subgroup Summary Report Custom Step

This example uses the **Designer** tab, which is available in SAS Studio Analyst and SAS Studio Engineer, to create the custom step required for this report. The custom step includes these controls:

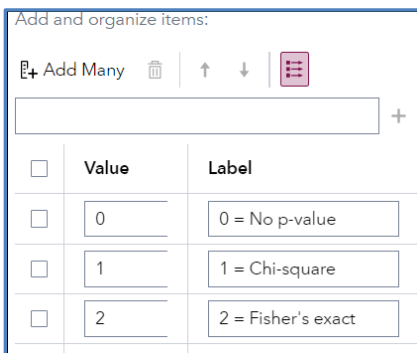
- three-page controls – the Data page, the Options page, and the About page
- section control – use this control to organize related controls on a page
- an input table control – use this control to select the data to use as input for the step. When the step is in a flow, this control is available when you click the input port.
- two column selector controls – use these controls to select columns from the input table. The first control is a multi-variables selection control for selecting the variables to be displayed in the report, and the second one is a single-variable selection control for selecting the BY group variable.

- two check box controls – one to control whether the report should be displayed by the BY group variable (if it is not selected, the summary report is generated for the whole sample), the other check box control is used to send errors or warnings about the report to the log.
- an output table control - use this control to specify the name and location of the output table.
- four drop-down list controls – use these controls to display report options for enhancing the report.

Step 1: Define the Controls

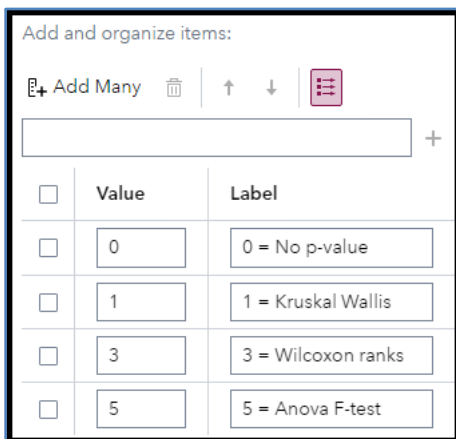
1. In the Steps pane, click  and select Custom step quick start. A new Custom Step.step tab opens.
2. In the Control Library, use the **+ Add Page** control to add three pages. Name the pages **Data**, **Options**, and **About**, in that order.
3. In the Control Library, scroll to the COMMON heading. Double-click or drag-and-drop the **Section** control to the canvas. Label the control *Demographic Table Summary Report*
4. In the Control Library, scroll to the DATA heading. Double-click **Input Table** to add an input table control to the canvas.
 - a. In the Properties: Input Table pane, provide values for the **ID** (*inputds*) and **Label** (*Input Data*) for the Input table.
 - b. Select the **Required** check box. The other parameters are optional.
5. In the Control Library, scroll to the DATA heading. Double-click **Column Selector** to add a column selector control to the canvas.
 - a. Modify these properties in the Properties: Column Selector pane.
 - i. In the **ID** field, enter *reportvars*
 - ii. In the **Label** field, enter *Report variables:*
 - iii. From the **Link to input table** drop-down list, select Input table (*inputds*).
 - iv. In the **Column type** field, select *All types*.
 - v. In the **Minimum columns** field, enter 1.
6. In the Control Library, scroll to the COMMON heading. Double-click or drag-and-drop the **Check Box** control to the canvas.
 - a. Modify these properties in the Properties: Check Box pane.
 - i. In the **ID** field, enter *byVar_flg*.
 - ii. Label the control *Display report by group variable*
 - iii. Select the **Checked by default** check box.
7. In the Control Library, scroll to the DATA heading. Double-click **Column Selector** to add a column selector control to the canvas.
 - a. Modify these properties in the Properties: Column Selector pane.
 - i. In the **ID** field, enter *byVar*
 - ii. In the **Label** field, enter *Report group variable:*
 - iii. From the **Link to input table** drop-down list, select Input table (*inputds*).
 - iv. In the **Column type** field, select *All types*.

- v. In the **Minimum columns** field, enter 1.
 - vi. In the **Maximum columns** field, enter 1.
 - vii. Scroll down to the Dependencies section. In the **Visibility** box, enter "\$byvar_flg". This creates a dependency between the check box and column selector controls to control whether the report is displayed by the BY group variable.
8. Navigate to the Options page.
9. In the Control Library, scroll to the COMMON heading. Double-click or drag-and-drop the **Drop-down List** control to the canvas.
- a. Modify these properties in the Properties: Drop-down List pane.
 - i. In the **ID** field, enter *pvalue_catgvar*
 - ii. In the **Label** field, enter *P-value option (categorical or binary variables)*:
 - iii. Leave the **Required** check box deselected.
 - iv. In the **Determine where values come from** field, select the radio button labeled **Static list**.
 - v. In the **Add and organize items** field, click **Add Many**. A list selection window opens. Enter the following values on each line.
 - o 0 = No p-value
 - o 1 = Chi-square
 - o 2 = Fisher's exact
 - vi. Click **OK** to close the list Items window.
 - vii. In the **Add and organize items** field, make the following adjustments to the list items. Enter the value for each option in the **Value** column for each **Label** field. This value is what is passed and processed by the SAS program when a selection is made.



- viii. In the **Default item** field, select 1=Chi-square as the default.
10. In the Control Library, scroll to the COMMON heading. Double-click or drag-and-drop the **Drop-down List** control to the canvas.
- a. Modify these properties in the Properties: Drop-down List pane.
 - i. In the **ID** field, enter *pvalue_contvar*
 - ii. In the **Label** field, enter *P-value option (continuous or date variables)*:
 - iii. Leave the **Required** check box deselected.

- iv. In the **Determine where values come from** field, select the radio button labeled **Static list**.
- v. In the **Add and organize items** field, click **Add Many**. A list selection window is opened. Enter the following values on each line:
 - o 0 = No p-value
 - o 1 = Kruskal Wallis
 - o 3 = Wilcoxon ranks
 - o 5 = Anova F-test
- vi. Click **OK** to close the list Items window.
- vii. In the **Add and organize items** field, make the following adjustments to the list items



- viii. In the **Default item** field, select 1=Kruskal Wallis as the default.
11. In the Control Library, scroll to the DATA heading. Double-click **Output Table** to add an output table control to the canvas.
 - a. In the Properties: Output Table pane, enter *Specify the report output table;* as the label.
 12. Repeat Step 10 for the remaining two drop-down list controls. In the Control Library, scroll to the COMMON heading. Double-click or drag-and-drop the **Drop-down List** control to the canvas. Make the necessary adjustments to the list items.
 - a. Modify these properties in the Properties: Drop-down List pane.
 - i. In the **ID** field, enter *report_odspath*, *report_orientation*, respectively.
 - ii. In the **Label** field, enter *Report output type*, *Report orientation*, respectively.
 13. In the Control Library, scroll to the COMMON heading. Double-click or drag-and-drop the **Check Box** control to the canvas.
 - a. Modify these properties in the Properties: Check Box pane.
 - i. In the **ID** field, enter *debug_flg*
 - ii. Label the control *Display report debugging information to the log*
 - iii. Do not select the **Checked by default** check box.

Step 2: Adding Your SAS Code

The next step is to add the SAS code for the Demographic Table and Subgroup Summary Report.

1. On the **Program** tab, add the following %TableN SAS code that can be retrieved via this link: <https://communities.sas.com/t5/SAS-Communities-Library/Demographic-Table-and-Subgroup-Summary-Macro-TABLEN/ta-p/634030>
2. The driver macro for the %TableN macro is provided in Appendix A.

Step 3: Save the Step

To save this custom step, click **Save as**. Though you can save the step to any location that you prefer, it is recommended that you save this step to a folder in SAS Content. Enter *Demographic Table and Subgroup Summary Report* as the name of the step.

Step 4: Add the Custom Step to a Studio Flow

To add the custom step to a new studio flow, perform the following:

1. From the main menu, select **New** → **Flow**. A blank flow opens.
2. To add the custom step to the flow, drag and drop the *Demographic Table and Subgroup Summary Report* from the Steps pane.

To add the Basic Example step to the flow, drag and drop the Basic Example step from the Steps pane. (See Figure 2.) At the right of the canvas, the user can see part of the user interface defined using the step **Designer** UI.

Step 5: Running the Custom Step as a Stand-alone

To save this custom step, click **Save as**. Though you can save the step to any location that you prefer, it is recommended to save the custom step to your SAS Server or SAS Content folder. [Note: The SAS code used for this custom step accepts SAS data sets and CAS tables as input data.]

Figures 4 and 5 display the custom step when it is finalized and the output of the custom step when it is executed as a stand-alone step.

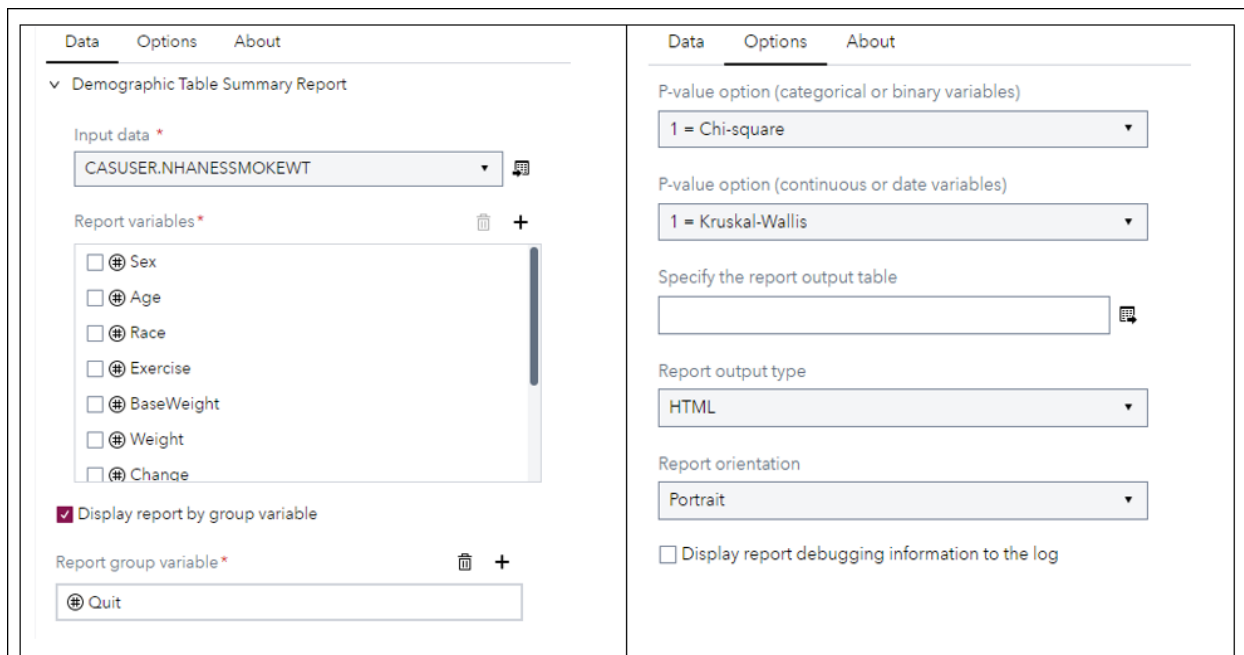


Figure 4. User Interface for Demographic Table and Subgroup Summary Report

| | 0 (N=1282) | 1 (N=464) | Total (N=1746) | P-value |
|--|---------------|--------------|-------------------|---------|
| Whether an individual quit smoking between the initial and follow-up interviews, 1=Yes, 0 otherwise | | | | |
| Level of daily activity, with values 0, 1, and 2, n (%) | | | | 0.26171 |
| 0 | 583 (45.5%) | 193 (41.6%) | 776 (44.4%) | |
| 1 | 577 (45.0%) | 218 (47.0%) | 795 (45.5%) | |
| 2 | 122 (9.5%) | 53 (11.4%) | 175 (10.0%) | |
| Age in 1971 | | | | <.00012 |
| N | 1282 | 464 | 1746 | |
| Mean (SD) | 42.7 (11.80) | 46.3 (12.53) | 43.6 (12.10) | |
| Median | 42.0 | 46.5 | 43.0 | |
| Range | 25.0, 72.0 | 25.0, 74.0 | 25.0, 74.0 | |
| Weight in kilograms in 1971 | | | | 0.01552 |
| N | 1282 | 464 | 1746 | |
| Mean (SD) | 70.5 (15.53) | 72.8 (16.96) | 71.1 (15.95) | |
| Median | 68.7 | 71.3 | 69.6 | |
| Range | 40.8, 169.2 | 36.2, 180.5 | 36.2, 180.5 | |
| Difference in weight at the follow-up and baseline interviews (measured in kilograms) | | | | <.00012 |
| N | 1242 | 437 | 1679 | |
| Mean (SD) | 2.1 (7.34) | 4.5 (8.71) | 2.7 (7.79) | |

Figure 5. Demographic Table and Subgroup Summary Report

USING CUSTOM STEP TO PERFORM PROPENSITY SCORE MATCH ANALYSIS

For this example, I used a custom step to create the code snippet and user interface for the propensity score match analysis. Propensity scoring method is used in outcomes and epidemiological research for sample selection by matching and analysis of causal effect by stratification. For a detailed explanation of propensity score matching, consult Rosenbaum (2010) and Rosenbaum and Rubin (1983). [Note: The code snippet used for creating this custom step requires CAS tables as input data. The step generates errors when tables in other formats are passed as input tables to the step.]

The screenshot displays the SAS Studio interface for a custom step titled 'Single cohort analysis'. The configuration pane on the left includes the following settings:

- Analysis data for propensity score model: CASUSER.NHANESMOKEWT
- Matching variable for propensity score model: Quit
- Event value for the matching variable: 1
- Patient/Subject ID: _obsid_
- Variables for propensity score model: Sex, Age, Race, Education, Exercise, BaseWeight, Weight (all selected)
- Impute values for variables with missing values: (unchecked)
- Output CAS table for propensity score matched data: CASUSER.testcohortmatch

The results pane on the right displays a table titled 'Characteristics of patients before and after propensity score matching'. The table compares characteristics between 'Before PS-matching' and 'After PS-matching' for Controls and Treated groups, including Standardized Mean Differences (SME).

| Characteristic | Level | Before PS-matching | | | After PS-matching | | |
|------------------|------------------|--------------------|----------------|--------|-------------------|----------------|--------|
| | | Controls | Treated | SME | Controls | Treated | SME |
| Baseweight | Baseweight | 70.51 (15.528) | 72.79 (16.960) | 0.133 | 71.36 (14.775) | 71.95 (15.464) | 0.039 |
| Perday | Perday | 21.25 (11.585) | 18.71 (12.208) | -0.217 | 18.28 (9.804) | 18.80 (12.438) | 0.043 |
| Yearssmoke | Yearssmoke | 23.98 (11.752) | 26.27 (12.842) | 0.159 | 26.43 (12.226) | 25.92 (12.838) | -0.042 |
| Logit Prop Score | Logit Prop Score | | | 0.717 | | | 0.011 |
| Prop Score | Prop Score | | | 0.710 | | | 0.014 |
| Sex | 0 | 607 (47.35) | 257 (55.39) | 0.160 | 201 (51.94) | 211 (54.52) | 0.052 |
| | 1 | 675 (52.65) | 207 (44.61) | | 186 (48.06) | 176 (45.48) | |
| Age | Age | 42.68 (11.798) | 46.27 (12.528) | 0.282 | 46.38 (11.974) | 45.80 (12.230) | -0.048 |
| Race | 0 | 1093 (85.26) | 418 (90.09) | 0.177 | 348 (89.92) | 351 (90.70) | 0.024 |
| | 1 | 189 (14.74) | 46 (9.91) | | 39 (10.08) | 36 (9.30) | |
| Education | 1 | 218 (18.15) | 93 (21.73) | 0.197 | 90 (23.26) | 77 (19.90) | 0.112 |
| | 2 | 273 (22.73) | 78 (18.22) | | 59 (15.25) | 71 (18.35) | |
| | 3 | 495 (41.22) | 164 (38.32) | | 152 (39.28) | 152 (39.28) | |
| | 4 | 96 (7.99) | 30 (7.01) | | 31 (8.01) | 28 (7.24) | |
| | 5 | 119 (9.91) | 63 (14.72) | | 55 (14.21) | 59 (15.25) | |
| Exercise | 0 | 267 (20.83) | 76 (16.38) | 0.117 | 58 (14.99) | 60 (15.50) | 0.015 |
| | 1 | 532 (41.50) | 198 (42.67) | | 175 (45.22) | 173 (44.70) | |
| | 2 | 483 (37.68) | 190 (40.95) | | 154 (39.79) | 154 (39.79) | |
| Weight | Weight | 72.50 (15.627) | 76.84 (16.978) | 0.280 | 75.15 (15.768) | 76.07 (16.494) | 0.056 |
| Change | Change | 2.13 (7.342) | 4.50 (8.710) | 0.313 | 3.80 (7.893) | 4.12 (8.365) | 0.040 |
| Activity | 0 | 583 (45.48) | 193 (41.59) | 0.088 | 164 (42.38) | 167 (43.15) | 0.034 |
| | 1 | 577 (45.01) | 218 (46.98) | | 180 (46.51) | 181 (46.77) | |
| | 2 | 122 (9.52) | 53 (11.42) | | 43 (11.11) | 39 (10.08) | |

Figure 6. Custom Step for Propensity Score Match Analysis

BEST PRACTICES FOR CREATING CUSTOM STEP

- Provide the description and key information about the custom step in the **About** tab to inform users about the capabilities and limitations of the step.
- Since a custom step can be executed in stand-alone mode or as part of a SAS Studio flow, first test the step in the stand-alone mode and fix any bugs or errors that are found before adding the step to a SAS Studio flow.
- Always check the log for errors, warnings, and resolution of macro variable names.
- Use port details to control the output columns.
- When applicable, use the starter custom step templates that are shipped with SAS Studio so that you don't have to start from scratch.
- Be part of the Git repository team contributors to showcase your work and share your custom steps with other users. The repository already contains several custom steps published by SAS staff.

CONCLUSION

SAS Studio Custom Step is a feature in SAS Studio that enables you to create custom code snippets that can be reused in your SAS programs. These snippets, known as custom steps, can help save time and streamline your workflow by automating repetitive tasks.

REFERENCES

Myers, Jeff. "Demographic Table and Subgroup Summary Macro %TABLEN". Accessed March 20,2023. Available at <https://communities.sas.com/t5/SAS-Communities-Library/Demographic-Table-and-Subgroup-Summary-Macro-TABLEN/ta-p/634030>.

Rosenbaum, P. R. (2010). Design of Observational Studies. New York: Springer-Verlag.

Rosenbaum, P. R., and Rubin, D. B. (1983). "The Central Role of the Propensity Score in Observational Studies for Causal Effects." Biometrika 70:41–55.

SAS Institute Inc. 2023. SAS® Studio: Working with Custom Steps. Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

This author would like to thank Heather Weinstein and Jim Box with the editing of the paper. Any mistakes and omissions are solely the responsibility of the author.

CONTACT INFORMATION

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

David Olaley
SAS Institute
David.Olaley@sas.com

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

APPENDIX A

The driver macro for the %TABLEN macro program for executing the demographic table and subgroup summary report:

```
*****;
*Macro to get user-supplied values*;
*****;

%macro getvarList(_varnameSelector=);
  %global _varlist;
  %let _varlist=;

  %do j = 1 %to  &&_varnameSelector._count;
    %let _varlist =  &_varlist. &&_varnameSelector._&j._name;
  %end;
  %put &=_varlist;
%mend;

%macro tablen_DemogReport;

  %if (%str(&inputds.) eq ) %then %goto ERROREXIT;

  %if %upcase(%scan(%bquote(&inputds.), 1)) ne WORK |
    %upcase(%scan(%bquote(&inputds.), 1)) ne SASUSER %then %do;

    data work.Table1Report;
      set &inputds.;
    run;

    %let inputds = %str(work.Table1Report);
  %end;

  %if (%bquote(&reportds) eq ) %then %let reportds = %str(reportTableDemog);

  %if (&debug_flg eq ) %then %let debug_flg=0;

  **Parse user-selected variables for target cohort**;

  %getvarList(_varnameSelector=reportvars);
  %let reportvars = %str(&_varlist);
  %put &=reportvars;

  %if %eval(&byVar_flg = 1) %then
  %let reportVars=%sysfunc(tranwrd(%str(&reportVars.), &byVar,  ));

  ** Get the variable contents information**;
  proc sql noprint;

  %if %sysfunc(exist(work.content_inputds)) %then %do;
    drop table work.content_inputds;
  %end;

  %if %sysfunc(exist(work.pvaluelist)) %then %do;
    drop table work.pvaluelist;
  %end;
quit;
```

```

%let _rptvars_ =;
%let pvalue_list =;
%let vartype_list =;

proc contents data=&inputds.(keep=&reportvars &byvar)
out=work.content_inputds(keep=name type) noprint;
run;

proc sql noprint;
  select count(*), name
  INTO :NVAR$ trimmed, :_rptvars_ separated by ' '
  FROM work.content_inputds;

  create table work.pvaluelist
    (name char(32), unique_value num);
quit;

%do j=1 %to &nvars;

  %let unique_value =;
  %let rptvar = %sysfunc(kscan(%bquote(&_rptvars_), &j, ' '));
  %let rptvar = &rptvar;

  proc sql noprint;
  insert into work.pvaluelist
  select "&rptvar" as name, count(distinct &rptvar.) as unique_value
  from &inputds(Keep=&rptvar);
  quit;
%end;

data work.content_inputds;
  merge
    work.content_inputds
    work.pvaluelist ;
  by name;

  length pvalue_list 8;
  if (type = 2) | (type=1 and unique_value le 4) then do;
    type=2; pvalue_list=&pvalue_catgvar.;
  end;
  else do;
    type = 1; pvalue_list=&pvalue_contvar.;
  end;
run;

proc sql noprint;
  select kstrip(name), type, pvalue_list
  into :_rptvars_ separated by ' ',
       :vartype_list separated by ' ',
       :pvalue_list separated by ' '
  from work.content_inputds;
quit;

%if %eval(&byVar_flg = 0) %then %do;
%tablen(
  DATA = &inputds.

```

```

        ,VAR = %str(&_rptvars_)
        ,TYPE= %str(&vartype_list)
    );
%end;

%if %eval(&byVar_flg = 1) %then %do;
%tablen(
    DATA = &inputds.
    ,VAR = %str(&_rptvars_)
    ,TYPE= %str(&vartype_list)
    ,BY = &byVar
    ,PVALS = %str(&pvalue_list)
    );
%end;

** Perform Clean up**;
%let inputds=;
%let reportvars=;
%let byVar_flg=;
%let byVar=;
%let pvalue_catgvar=;
%let pvalue_contvar=;
%let reportds =;
%let debug_flg=;
%let _rptvars_=;
%let vartype_list=;
%let pvalue_list=;

%ERROREXIT:

%mend tablen_DemogReport;

```