SESUG Paper 214-2023

Methods and Tools for Publishing at SESUG and Beyond—Basic Techniques

James Blum, University of North Carolina Wilmington
Jonathan Duggins, North Carolina State University

Abstract

For programmers, publishing papers or other documents may be a frustrating task. We show some methods of applying programming in LaTeX and SAS to create styles and structures that can be easily reused and revised, with the added benefit of actually looking good. We build styles for various paper elements including boxes/titles for code, output and logs (and other items). LaTeX tools for numbering sections, programs, output, and other elements inside the document are demonstrated along with making cross-references to such items which automatically update when items are moved or removed. An introduction to the ODS TAGSETS.COLORLATEX destination and the resulting sastable object is also provided. All LaTeX implementations are covered using the Overleaf platform, but other LaTeX environments work as well.

Introduction

In this paper, we introduce concepts for publishing documents about SAS programming using LaTeX. If you are not familiar with LaTeX, it is a document preparation system that is conducive to scientific writing. To get the most out of this paper, it is helpful to be familiar with the basics of writing and compiling LaTeX documents. For readers just getting started with LaTeX, we suggest starting with the LaTeX Wikibook: https://en.wikibooks.org/wiki/LaTeX. While we focus on publishing a SESUG paper, the techniques covered are extendable to other types of articles and even to books. This paper presents fundamental methods and tools, both in LaTeX and SAS, for simplifying the publishing process. More advanced topics are available in the companion paper "Methods and Tools for Publishing at SESUG and Beyond-Advanced Techniques".

Given that this is a paper about how to write a paper, our structure for it is somewhat unusual because the final output is a complete paper in PDF form rather than a single table or graph. So, be aware that there will be a mixture of SAS and LaTeX code as we progress through the examples.

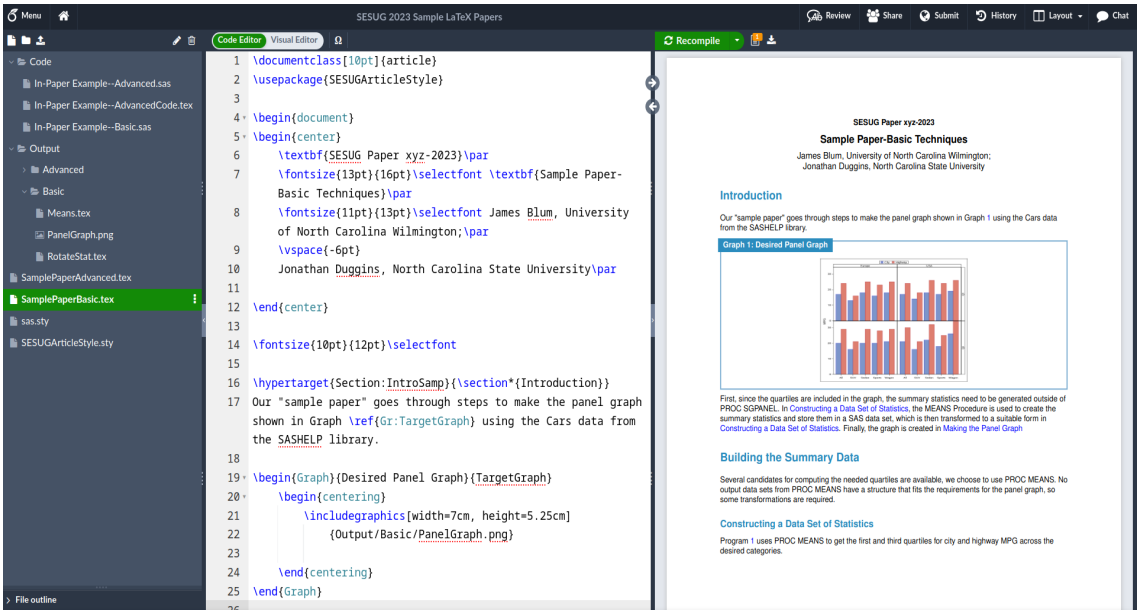
Sections that follow go through various aspects of the LaTeX code used to create the sample paper given in Appendix A–you should jump there now and review it. The sample paper and all associated files to create it can be downloaded at https://jonathanduggins.com/conferences and, for those who have not used Overleaf previously, further details on setting up the project are given in Appendix B.

The download includes style files along with the sample paper, and those style files can be used without modification. However, we will cover some of those details so that you can make changes if you wish to do so. Our discussion of the LaTeX code is ordered to match the contents of the sample paper. So, after a look at the Overleaf environment, we start our LaTeX coding by building the sample paper title, authors, and Sample Paper Introduction.

A Look at the Overleaf LaTeX Environment

First, we start with a brief look at the Overleaf environment used for building the sample paper. Overleaf is not required, and almost any LaTeX environment will work, but you may have to load some LaTeX packages that Overleaf loads by default. Overleaf is a collaborative environment, with the number of collaborators and related features controlled by your subscription level–the base subscription level is free and allows a single collaborator on each project. Figure 1 shows the Overleaf environment used to build the sample paper.

Figure 1: Introduction to the Overleaf Environment



The left panel is the project directory tree, which contains various folders and files, and these are what you will see if you download the sample project we provided and upload it to Overleaf. For the next few sections, we focus on the files with "Basic" as part of their name or folder structure; however, the environment in Figure 1 (and the download) contains the sample papers for both the basic and advanced techniques.

The middle panel is the LaTeX code panel–though it can display information in other file types–and displays the file highlighted in the directory tree in the left panel. Line numbers, autofill, code hints, and code highlighting are standard features in this window and, depending on your subscription level, it also supports tracked changes and comments.

The right panel is the pdf rendering of the file most recently compiled (presuming it is a .tex file that can be compiled successfully). At the top left of that panel is the Recompile button, which compiles the currently selected file (assuming that is appropriate). The button next to Recompile switches you between the pdf rendering and the log for the compilation. The third button allows you to download the compiled result as a pdf file.

Various environmental options are available. For example, you can choose to display the pdf in a separate browser tab (or window), leaving the code and file tree in another–very convenient when you have multiple monitors available. Code highlighting is changeable, as is the LaTeX compiler and a host of other options. For more details on this, other Overleaf features, and various LaTeX support, see the Overleaf documentation.

Next, we take a more detailed look at the LaTeX code you can see in the center panel Figure 1

Starting the Paper in LaTeX

LaTeX Code 1 shows the opening code for the paper, through the title and author section.

LaTeX Code 1: LaTeX Preamble Plus Code for Title and Authors

```
\documentclass[10pt]{article}❶
\usepackage{SESUGArticleStyle}❷

\begin{document}❸
\begin{center}❹
    \textbf{SESUG Paper xyz-2023}\par❺
    \fontsize{13pt}{16pt}\selectfont❻
       \textbf{Sample Paper-Basic Techniques}\par
    \fontsize{11pt}{13pt}\selectfont James Blum,
        University of North Carolina Wilmington;\par
    \vspace{-6pt}❼
    Jonathan Duggins, North Carolina State University\par

\end{center}❹
```

❶ The DOCUMENTCLASS command chooses the document type, potentially with options. All LaTeX commands begin with the \ character. Items in square brackets are optional, those in braces are required (multiple arguments are separated by commas).

❷ The USEPACKAGE command names a package (perhaps with options) that the LaTeX file is expected to use. Here, the reference SESUGArticleStyle points to the LaTeX style file with the same name and an .STY extension shown in the file tree in Figure 1.

❸ Everything that appears in the LaTeX document goes between the BEGIN{Document} and END{Document} commands. You can think of BEGIN in the same way as a SAS DO group, it must have a corresponding END.

❹ The first part of any SESUG paper is the paper number, title, and author set, which the template has centered on the page. To achieve this, a BEGIN/END block is placed around this section using the Center parameter.

❺ The TEXTBF command is for boldface text, PAR forces a paragraph/line break. Line breaks in the editor do not force a line break in the output file.

❻ The FONTSIZE command sets both the font size (first argument) and the distance between lines in paragraph breaks (second argument). SELECTFONT is required to apply this setting to the current font.

❼ The second argument to FONTSIZE has some built-in safety features–it will not let lines collide even if you set it to zero. Here, we use VSPACE with a negative argument to reduce the space between the two author lines.

In the DOUCMENTCLASS command, the class you choose defines the standard layout for your document. The most commonly used document classes in LaTeX are:

- Article – for relatively short papers like this one.
- Report – for longer reports or short books.
- Book – for books.
- Beamer – for presentation slides.

The use of the optional argument of 10pt font size is redundant in this example, it is there merely to illustrate how to use it. First, that is the default for the Article class. Second, as you have seen, we take control of font sizing inline with the FONTSIZE command.

We will go over various elements of the SESUGArticleStyle.sty file as we progress, starting with LaTeX code 3. For now, know that style files are typically used to reduce the size of the LaTeX preamble by storing needed package loads, commands, options, and other LaTeX code in a separate file. (The LaTeX preamble is the set of statements prior to the \BEGIN{document} statement).

Sample Paper Introduction

Next up, we take a look at some of the elements used in writing the introduction to the sample paper, including: writing section headings, using boxes, inserting an image, and inserting hyperlinks. LaTeX Code 2 picks up where LaTeX Code 1 left off, and provides a portion of the introduction from the sample paper.

LaTeX Code 2: Elements of the Sample Paper Introduction

```
\fontsize{10pt}{12pt}\selectfont❶

\hypertarget{Section:IntroSamp}{\section*{Introduction}}❷
Our "sample paper" goes through steps to make the panel graph shown in
Graph \ref{Graph:TargetGraph}❸ using the Cars data from the
SASHELP library.

\begin{Graph}{Desired Panel Graph}{TargetGraph}❹
    \begin{centering}❺
        \includegraphics[width=7cm, height=5.25cm]
            {Output/Basic/PanelGraph.png}❻

    \end{centering}❺
\end{Graph}❹


First, since the quartiles are included in the graph, the summary
statistics need to be generated outside of PROC SGPANEL. In
\hyperlink{SubSection:BasicStats}{Constructing a Data Set of
Statistics}❻, ...
```

❶ We return to the 10pt sizing for the remainder of the paper

❷ HYPERTARGET creates a target for hyperlinking. The first argument is a label used to identify the target when linking to it, the second argument is the text to display. The text in the second argument becomes a section heading via use of the SECTION command, the * suppresses numbering of the section.

❸ The REF command can be used to link to labeled items in the file. This links the the Graph item below and uses its number as the link text.

❹ The Graph environment is not a standard LaTeX environment, rather it has been defined via extending the package TColorBox. These definitions are in the style file, and are discussed below.

❺ In this instance we center the image inside the Graph box using the Centering environment. We use Centering instead of Center to avoid extra, vertical spacing that can be generated by the Center environment.

❻ Inside the graph box, an image is inserted via the INCLUDEGRAPHICS command. Though the arguments to set the width and height are optional, we advocate taking direct control over them as the default is the actual image size. The required argument is the file name and path for the image. In Overleaf, these are Linux paths, so use the / as a separator and mind your casing.

❼ HYPERLINK creates a link to an item defined by HYPERTARGET. The first argument is the label of the target, the second is the text to associate with the link. Styling of the link is located in our style file, more details on that below.

Since the SESUG template uses unnumbered section headings, we choose to create section headings as part of a HYPERTARGET reference and link to them with HYPERREF, though other methods are possible. The REF command, used to link to the first Graph box here, can only reference and link to numbered

objects. In general, we recommend creating link targets in the form: *prefix:name*, to make it easier to construct and reference names while avoiding duplication. In order for these linking commands to work, the HyperRef package must be loaded. That can be done as part of the preamble of the document, but we choose to do it in a style file, SESUGArticleStyle referenced in LaTeX Code 1. Several of the items included in LaTeX Code 2 (and the remainder of the paper) depend on settings in SESUGArticleStyle.sty, so we move to that file in LaTeX Code 3 to begin highlighting its features.

LaTeX Code 3: A First Look at Some Elements of the Style File

```
\usepackage{hyperref}❶
\hypersetup{colorlinks=true, linkcolor=blue,
    filecolor=magenta, urlcolor=cyan}❷
\urlstyle{same}❸
```

❶ The HyperRef package allows for hyperlinking in the document.

❷ HYPERSETUP sets options for hyperlinks. ColorLinks=True colors the link text (default is to put a box around links). LinkColor= applies to links internal to the document, FileColor= applies to linked files, UrlColor= applies to external urls.

❸ URLSTYLE{Same} makes the link font style the same as the current text style (default is a monospace font).

All of the commands in LaTeX Code 3 could have been placed in the LaTeX preamble shown in LaTeX Code 1 (between DOCUMENTCLASS and BEGIN{Document}), but it is more efficient to place these into a style file that can be used with multiple documents. Other items that could have been in the preamble but are better placed in the style file include general formatting options to match the SESUG article format. These are effectively global settings appearing only in the style file, and are not altered in the paper. They are shown in LaTeX Code 12 in Appendix C.

Other items defined in the style file include custom colors, which are used in headings and boxes, like the Graph box. LaTeX Code 4 shows two excerpts from the style file that set up colors and apply them, and other styles, to the headings.

LaTeX Code 4: Colors and Section Headers

```
\usepackage{xcolor}❶
\definecolor{prgHeadBlue}{RGB}{43,140,190}❷
\definecolor{prgBrdBlue}{RGB}{116,169,207}❷
\definecolor{prgBackBlue}{RGB}{241,238,246}❷
.
.
.
\usepackage{titlesec}❸

\titleformat*{\section}{\Large\bfseries\sffamily\color{prgHeadBlue}}❹
\titleformat*{\subsection}{\large\bfseries\sffamily\color{prgHeadBlue}}❹
```

❶ The XColor package allows for creation of custom color definitions.

❷ DEFINECOLOR has three required arguments: a name for the defined color, a color model, and a color definition according to that color model. Here we use the RGB color model, with intensity defined on each channel in the range 0 to 255.

❸ The TitleSec package is loaded to allow for re-styling of section headers.

❹ TITLEFORMAT* is simplified title styling (TITLEFORMAT is more complex) and has two required arguments: which level of titling to target, and styles to apply. In the styling, we use the relative font sizing of Large for Section and large for SubSection, with a bold font (BFSERIES), our sans family again, with our prgHeadBlue custom color.

Many more colors are defined in our style file, and there is no limit on the number you can set up. LaTeX also has some pre-named colors; however, we recommend defining your own so that you know exactly what you are getting.

The Graph box uses these custom colors, LaTeX Code 5 shows this use and more details on how that box is created.

LaTeX Code 5: Colors and Section Headers

```
\usepackage[most]{tcolorbox}❶

\newtcbtheorem{Graph❷}{Graph❸}{
        enhanced,
        sharp corners,
        attach boxed title to top left={
            xshift=-1mm,
            yshift=-5mm,
            yshifttext=-1mm},
        boxed title style={
            sharp corners,
            size=small,
            colback=prgHeadBlue,
            colframe=prgHeadBlue},
        fonttitle=\bfseries,
        top=1.5em,
        colback=white,
        colframe=prgBrdBlue}❹
    {Gr❺}
```

❶ The TColorBox package allows you to define boxes like the ones used for the Graph in the sample paper introduction. The Most option leaves out a few associated packages that we do not need.

❷ The NEWTCBTHEOREM command creates a new theorem (this was made largely for math books) box. The first argument is the name of the box that is used in the BEGIN and END commands when requesting a box of this type.

❸ The second argument is the text used as a prefix/label in the title box. This does not have to match ❷, but it can, they play different roles and there is no conflict if the names are the same.

❹ The fourth argument contains the styling options you wish to apply to your box. A bit more detail is given below, and you can consult the TColorBox documentation for more information on styling options.

❺ The last argument is the prefix used in a REF command. This does not have to match ❷ or ❸, but it is permitted. Like this instance, many times we shorten this reference to save typing effort, especially for boxes that are used often.

When you download the style file, you will also find box definitions for Program, Output, and Log, along with others. Note that some of boxes have an optional counter argument for controlling how the numbering progresses. Numbering can be tied to section or subsection numbering if such numbering is active (not for SESUG papers), or numbering can be suppressed entirely. If no numbering option is given, as with our Graph box, the default numbering is sequential for each use of the object, hence the reason the Graph box in the sample paper introduction is numbered 1.

All boxes used in the sample paper use the same basic styling. As you can see, the boxes in this portion of the paper use other style choices such as the box color (blue vs purple) and corner shape. The styling options in TColorBox are vast, some of the styling options we use include:

- Enhanced–Permits enhancements like the title box.

- Attach Boxed Title To Top Left–Positions the title box. Default position for a Top Left title box has the bottom left corner of the title box at the same point as the top left corner of the bounding box. The Shift options move the box/text.
- Boxed Title Style–Styling for the title box. We have chosen sharp corners for the sample paper, rounded corners are default for both the title and bounding boxes. The title box is made up of a frame and background, and colors can be set for each. Default text color is white in the title box, so a fairly strong color is assumed for the background.
- Top–Margin from the top of the bounding box to items contained within. Other margins are changeable as well, and the spacing between the bounding box and items within also includes padding (which is also changeable).
- ColBack and ColFrame–The bounding box also has a frame and background, with each color settable.

Let's revisit a portion of LaTeX Code 2 to see how this Graph box was used.

LaTeX Code 6: Using the Graph Box

```
Our "sample paper" goes through steps to make the panel graph shown in
Graph \ref{Gr:TargetGraph}❸ using the Cars data from the
SASHELP library.

\begin{Graph}❶{Desired Panel Graph❷}{TargetGraph❸}
    \begin{centering}
        \includegraphics[width=7cm, height=5.25cm]
            {Output/Basic/PanelGraph.png}

    \end{centering}
\end{Graph}❶
```

- The BEGIN and END commands refer to the name Graph given as the first argument to NEWTCBTHEOREM in LaTeX Code 5.
- Two arguments are required with BEGIN{Graph}, the first is the title text for the box–which appears after the automatically generated text "Graph 1: ".
- The second argument is the label for the box. This is why the REF command in the prose before the Graph box uses Gr:TargetGraph as its argument. Gr is the prefix created in the final argument to NEWTCBTHEOREM in LaTeX Code 5, TargetGraph is the name chosen for this box.

This feels like a lot of work just to get a simple introduction written, but most of the work for the entire paper has been accomplished. Global styles are set and known, and use of boxes is almost identical across the box definitions. We still need to learn how to put SAS code in the box, annotate it with call outs, and tie those call outs to comments.

Creating a SAS Program Box with Call Outs

Creating a SAS Program Box

LaTeX Code 7 shows the code to build the Program 1 box in the sample paper.

LaTeX Code 7: Creating a SAS Code Box

```
\begin{Program}{Using PROC MEANS to Compute Quartiles}{Means}❶
    \begin{lstlisting}❷[language=SAS, escapeinside=`~]❸
ods select none;`\CallOut{1}~❹
proc means data=sashelp.cars;
  where type not in ('Hybrid','Truck') and origin ne 'Asia';`\CallOut{2}~❹
  class origin type;`\CallOut{3}~❹
  var mpg:;`\CallOut{4}~❹
  output out=summary q1=City_Q1 Highway_Q1
         q3=City_Q3 Highway_Q3;`\CallOut{5}~❹
run;

ods select all;
proc print data=summary;
run;
    \end{lstlisting}❷
\end{Program}❶
```

❶ Setting up a Program box works much like the Graph box demonstrated in Latex Code 6. The BEGIN command refers to the Program box in the first argument, provides a title in the second, and labels it in the third. The END command refers back to the Program box.

❷ The LstListing environment is a verbatim environment, all text between the BEGIN and END commands is ignored by the LaTeX compiler and displays exactly as written, perfect for displaying code. Some options are set for this environment in the style file, and are shown in Latex Code 13 in Appendix C.

❸ The EscapeInside option sets starting and ending characters that wrap sections of text you do want to be interpreted by the LaTeX compiler. Language=SAS does not do much, but is there for potential future updates in SAS code highlighting in LstListing.

❹ The primary items inside an LstListing environment we need the LaTeX compiler to interpret are the requests for call outs in the SAS code; so, we use the CallOut function inside the escape character set. The CallOut function is a custom function we have defined, its required argument is the number to be displayed, and it permits setting a color as an optional argument. See Latex Code 14 in Appendix C for more details on this custom function.

Presuming we have built our program box, copied our SAS code into the LstListing section, and added our call outs; now we must add in our list of comments that are associated with each call out. LaTeX supports unordered lists (ITEMIZE) and ordered lists (ENUMERATE) natively, but we have modified ENUMERATE to number with the same Dingbats symbols from the PiFont package used by the CallOut function.

Creating a Call Out List

LaTeX Code 8 shows how the list of comments associated with the call outs in Program 1 are built.

LaTeX Code 8: Creating a Call Out List

```
\begin{enumerateCallOut}❶
    \item❷ There is no need to see the standard output from the
    MEANS Procedure, so it is suppressed.
    \item❷ The WHERE statement reduces the data to the desired categories.
    It is necessary to do this for Type at this step so that unwanted
    observations do not affect the calculation of the statistic.
    \item❷ Origin is one of the panel variables, and Type is the charting
    variable within each panel, so stratification on each is necessary.
    \item❷ Recall, the : is a wildcard for an arbitrary suffix, so this
    includes both MPG variables.
    \item❷ In the OUTPUT statement, each statistic keyword has two variable
    names listed after it as there are two analysis variables in the
    VAR statement.
\end{enumerateCallOut}
```

❶ The EnumerateCallOut environment is an instance of the Enumerate environment we modified to number with the Wingdings symbols used for call outs. It has no required arguments, but does have one optional argument that sets the color of the numbers.

❷ Each item command sets forth a new list element numbered with the next number in the sequence.

The code for EnumerateCallOut is contained in the style file, but the details of how it is programmed are beyond the scope of this paper. Note that for both CallOut and EnumerateCallOut, 10 is the last/largest value permitted. Of course, if we were considering having more than 10 call outs in an example, we should probably consider simplifying that example, anyway.

Adding in Tabular Output

Now we want to add in the Output box and table that correspond to Program 1. First, we go over how to get a LaTeX version of the output from a SAS code submission, then we will put it into the paper.

Creating LaTeX Output Tables in SAS

We use ODS TAGSETS to create LaTeX tables from SAS output; specifically, ODS TAGSETS.COLORLATEX. Note, there are several TAGSETS that correspond to LaTeX, and all are pre-production, so the documentation is limited and the behavior is unpredictable at times. Nevertheless, TAGSETS.COLORLATEX works well for most of what we need. SAS Program 1 shows the actual code submitted to generate the output for Program 1.

SAS Program 1: Actual SAS Code for Program 1 Output

```
ods noproctitle;❶
ods tagsets.colorlatex
      file = "Means.tex"❷ (notop nobot)❸
      stylesheet = "SAS.sty" ❹
  ;
ods select none;
proc means data=sashelp.cars;
  where type not in ('Hybrid','Truck') and origin ne 'Asia';
  class origin type;
  var mpg:;
  output out=summary q1=City_Q1 Highway_Q1 q3=City_Q3 Highway_Q3;
run;

ods select all;
proc print data=summary;
run;
ods tagsets.colorlatex close;❷
```

❶ We generally avoid procedure titles in our output, so NOPROCTITLE is set at the outset. It is also a good practice to reset titles and footnotes here or earlier.

❷ When using TAGSETS.COLORLATEX, the ODS wrapper looks like it would for a PDF, RTF, or other output file. Make sure you have set your desired path so that Means.tex is delivered to an appropriate location.

❸ By default, TAGSETS.COLORLATEX generates a standalone LaTeX document, with any needed commands in the preamble or post document layout. Since we are inserting the LaTeX code into another document, we do not want this, so the NOTOP and NOBOT arguments remove it.

❹ Styling is inserted into the preamble, by default. We do want the styling but we do not want it there. STYLESHEET= is used to make a separate .STY file for the output that we then upload to Overleaf.

The SESUGArticleStyle.sty file contains the command \usepackage[color]{sas}, so our custom style file references the SAS-generated style file, making both active when the LaTeX code is compiled. Two other features of note: First, the ODS wrapper is placed around all of the code for Program 1, rather than just the portion that generates output (the PROC PRINT). This is a conscious choice for the sake of consistency–the code inserted into the document is that which is in the ODS wrapper. Second, the name of the output file is the same as the name of the Program and Output boxes it corresponds to. Again, this is a conscious choice to maintain consistency.

Note further that none of the naming/labeling includes numbers in our SAS or LaTeX code, even though boxes in the paper are numbered. One of the major advantages to using the automatic numbering features in LaTeX is that renumbering is then automatic when items are moved or removed. Using numbers in your internal names/labels will only create extra work and/or confusion if items need to be reordered later.

Adding LaTeX Output Tables to the Paper

Now we need to add the table generated by ODS TAGSETS.COLORLATEX into an Output box in the paper. As LaTeX Code 9 shows, it is actually relatively simple.

LaTeX Code 9: Putting a SAS Output Table into the LaTeX Document

```
\begin{Output}❶
    {\ref*{P:Means}:❷ Data Set from PROC MEANS OUTPUT Statement}{Means}❸
    \input{Output/Basic/Means}❹
\end{Output}❶
```

❶ We start by creating an Output box, which is similar to a Graph or Program box.

❷ The Output box is defined in the style file without numbering. So, in the first part of the title, we number it by using the REF command to reference the Program box with the code that generates it (we manually put in the colon). The * on the REF command suppresses the hyperlinking.

❸ The final argument is the name of the Output box, which, as noted above, we choose to match the name of the Program box and the .TEX file that contains the output table.

❹ We use the INPUT function to bring in the table. The table must be uploaded to the Overleaf environment, and the INPUT function gives the name and path to the file.

We choose to number Output boxes by referencing Program boxes so that the Output box number stays in direct correspondence to the Program box numbers. There are times when we display code that has no output (see Program 3 in the sample paper). There are also times where we may want to display output without a program, but we typically do that with a Graph, Figure, or Table box.

The * is used on the REF command for two reasons. First, the Output and Program boxes are typically in close proximity. Second, when hyperlinks are done in color, they are all done in color. So, blue hyperlink text on the blue background of a title box (or most other strong colors) will not work out so well. We could make a manual intervention for the color, but it is not worth the difficulty.

Since the NOTOP and NOBOT options are used for the file generated by TAGSETS.COLORLATEX, the tables are typically set up well for use with the LaTeX INPUT command. In general, INPUT brings in the whole LaTeX file, which can cause serious conflicts if it is a complete LaTeX document with a preamble. The SESUGArticleStyle.sty file also contains the command \usepackage{standalone} to offer extra protection against this issue–StandAlone restricts INPUT to the portion of the LaTeX file between \begin{document} and \end{document}, leaving the BEGIN and END commands out.

Modifications to LaTeX Output Tables

Program 2 is built much like Program 1, but its output has a couple of important differences. It is not a full listing of all output rows, and it has some call outs in the headers. The first change is done in the actual code, as shown in SAS Program 2

SAS Program 2: Actual SAS Code for Program 2 Output

```
ods tagsets.colorlatex file='RotateStat.tex' (notop nobot)
      stylesheet = "SAS.sty"
  ;
proc transpose data=summary
    out=summary2(drop = _type_ _label_ rename=(col1=MPG));
  where not missing(origin);
  by _type_ origin type;
  var city: highway:;
run;

proc print data=summary2(obs=12);
run;
ods tagsets.colorlatex close;
```

Beyond the ODS wrapper, the only difference between the code in Program 2 and SAS Program 2 is the OBS=12 option attached to the Summary2 data in PROC PRINT. So, when pasting in the code, we made the modification to eliminate the option and title the output with "Partial Listing". Of course, there is no harm in leaving the OBS= option in the code, but for novice users you will likely need to consider the relationship between the code displayed, the output the code will make, and limitations on output you display in the paper. In the paper "Methods and Tools for Publishing at SESUG and Beyond-Advanced

Techniques", we seek to automate the code and output insertion into the paper, and we give a more robust solution for limiting rows in an output table.

The addition of call outs to the headers in Output 2 is done manually in the RotateStat.tex file. LaTeX Code 10 shows a partial listing of the modified RotateStat.tex file, including two SASContents lines (which are not included in the LaTeX output) and the start of the table through the end of the header row.

LaTeX Code 10: Modified LaTeX Output Table

```
\sascontents[1]{The Print Procedure}
\sascontents[2]{Data Set WORK.SUMMARY2}
\begin{sastable}[c]{rlllr}\hline
   \multicolumn{1}{|S{header}{r}|}{Obs} &
   \multicolumn{1}{|S{header}{l}|}{Origin} &
   \multicolumn{1}{|S{header}{l}|}{Type \CallOut[OutOrange]{1}} &
   \multicolumn{1}{|S{header}{l}|}
        {{\textunderscore}NAME{\textunderscore} \CallOut[OutOrange]{2}} &
   \multicolumn{1}{|S{header}{r}|}{MPG}
\\\hline
```

So, we have gone in and added two CallOut commands, with the OutOrange color, one next to the Type header and another next to the _NAME_ header. One way or another, either in the SAS/ODS realm or on the LaTeX side, this is a manual intervention. Therefore, we recommend that you use it sparingly, if at all.

Adding in a SAS Log

Program 3 does not have a corresponding Output box, but its SAS log is shown. LaTeX Code 11 shows how the Log box is constructed for that program.

LaTeX Code 11: Creating a SAS Code Box

```
\begin{Log}{\ref*{P:FinalData}: Final Transformation of the Quartile Data}
    {FinalData}❶
      \begin{lstlisting}[escapeinside=`~]
 174        data summary2;
 175          set summary2;
 176          if missing(type) then type = 'All';
 177          Statistic = scan(_name_,2,'_');
 178          Category = scan(_name_,1,'_');
 179          drop _name_;
 180          length statistic category $8;
 `\aftergroup\warningcolor~❷
 WARNING: Length of character variable Statistic has already been set.
          Use the LENGTH statement as the very first statement in the
          DATA STEP to declare the length of a character variable.
 WARNING: Length of character variable Category has already been set.
          Use the LENGTH statement as the very first statement in the
          DATA STEP to declare the length of a character variable.
`\aftergroup\endwarningcolor~❷
 181        run;

`\aftergroup\notecolor~❷
 NOTE: There were 40 observations read from the data set WORK.SUMMARY2.
 NOTE: The data set WORK.SUMMARY2 has 40 observations and 5 variables.
 NOTE: DATA statement used (Total process time):
       real time           0.00 seconds
       cpu time            0.00 seconds
`\aftergroup\endnotecolor~❷
    \end{lstlisting}
\end{Log}
```

❶ The setup of a Log box mirrors that of an Output box, like the one built in LaTeX Code 9. The log text is a cut-and-paste operation from SAS to LaTeX in this case.

❷ To mimic coloring of text in the SAS log in our LstListing environment in LaTeX a special wrapper is defined in our custom style file and then applied here using the AFTERGROUP command. WarningColor, NoteColor, ErrorColor, along with an End for each are defined in our custom style file.

Defining changes in text styling (including color) in the LstListing environment is not trivial. The LaTeX command(s) given in the escape character set have to alter the default setting for the LstListing environment. AFTERGROUP is but one way to do this–the details are part of the custom style file and are beyond the scope of this paper.

Some Useful Graphics Options

To finish the sample paper, we need a Program box into which the PROC SGPANEL code is pasted, and an Output box that includes the panel graph image–both of which you should know how to do at this point. So, our final bit of LaTeX code looks at some recommendations for options used in SAS when we create graphics for publication.

SAS Program 3: Actual SAS Code for Program 4 Output

```
ods listing image_dpi=300;❶
ods graphics / reset❷ imagename='PanelGraph'❸ imagefmt=png❹;
proc sgpanel data=summary2;
  panelby origin statistic/ layout=lattice novarname;
  vbar type / group=category response=MPG groupdisplay=cluster;
  rowaxis label='MPG' offsetmin=0;
  colaxis display=(nolabel);
  keylegend / position=top title='';
run;
```

❶ All graphics are created (and uploaded to Overleaf) as image files, and are not directed to the file with the tabular LaTeX output. The resolution is set to 300 dpi so that the image can be reliably stretched without loss of quality.

❷ In the ODS GRAPHICS statement the RESET option (equivalent to RESET = ALL) sets all ODS graphics options back to their defaults. In particular, we want to make sure that the image name counter resets so that submissions of the program replace the image in our output folder.

❸ There are not many restrictions on our choice for the image name; however, we choose something which we also plan to use in the LaTeX code. If you look at the LaTeX code for the sample paper, you see that both the Program and Output boxes are also given the name PanelGraph.

❹ The default file type for images from ODS graphics is PNG, so we are simply being explicit here. PNG works well in LaTeX and is our default choice for all images, even those not generated by SAS.

We typically deliver our graphics output and tabular output to the same folder, which is chosen via the DLGCDIR function at the top of our SAS code. Of course, other structures are possible, but be sure you keep ease of uploading to Overleaf in mind when choosing a structure.

Conclusion

Hopefully, you now have enough information to create a technical document on SAS programming, such as a SESUG paper, using LaTeX. The ideas here are extendable, including to publishing a book. However, as documents get more complex, we prefer more automation and error control. If you expect to create some complex documents, check out our paper "Methods and Tools for Publishing at SESUG and Beyond-Advanced Techniques". It covers several useful extensions, including: Automating conversion of SAS code (including call outs) to TEX files, parsing a single TEX file of tabular output into separate boxes, and controlling some table sizing and styling without manual intervention.

Recommended Reading

Wikipedia. 2023. “LaTeX Wikibook.” https://en.wikibooks.org/w/index.php?title=LaTeX&stableid=4246546

Contact Information

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

James Blum, University of North Carolina Wilmington
blumj@uncw.edu
http://people.uncw.edu/blumj

Appendix A–Sample Paper

SESUG Paper xyz-2023

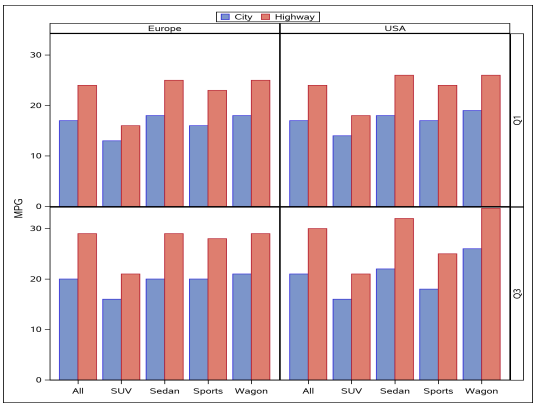
Sample Paper-Basic Techniques

James Blum, University of North Carolina Wilmington
Jonathan Duggins, North Carolina State University

Introduction

Our "sample paper" goes through steps to make the panel graph shown in Graph 1 using the Cars data from the SASHELP library.

Graph 1: Desired Panel Graph



First, since the quartiles are included in the graph, the summary statistics need to be generated outside of PROC SGPANEL. In Constructing a Data Set of Statistics, the MEANS Procedure is used to create the summary statistics and store them in a SAS data set, which is then transformed to a suitable form in Constructing a Data Set of Statistics. Finally, the graph is created in Making the Panel Graph

Building the Summary Data

Several candidates for computing the needed quartiles are available, we choose to use PROC MEANS. No output data sets from PROC MEANS have a structure that fits the requirements for the panel graph, so some transformations are required.

Constructing a Data Set of Statistics

Program 1 uses PROC MEANS to get the first and third quartiles for city and highway MPG across the desired categories.

Program 1: Using PROC MEANS to Compute Quartiles

```
ods select none;❶
proc means data=sashelp.cars;
  where type not in ('Hybrid','Truck') and origin ne 'Asia';❷
  class origin type;❸
  var mpg:;❹
  output out=summary q1=City_Q1 Highway_Q1 q3=City_Q3 Highway_Q3;❺
run;

ods select all;
proc print data=summary;
run;
```

❶ There is no need to see the standard output from the MEANS Procedure, so it is suppressed.

❷ The WHERE statement reduces the data to the desired categories. It is necessary to do this for Type at this step so that unwanted observations do not affect the calculation of the statistic.

❸ Origin is one of the panel variables, and Type is the charting variable within each panel, so stratification on each is necessary.

❹ Recall, the : is a wildcard for an arbitrary suffix, so this includes both MPG variables.

❺ In the OUTPUT statement, each statistic keyword has two variable names listed after it as there are two analysis variables in the VAR statement.

Looking at Output 1, we can see that the data set generated by the OUTPUT statement includes marginal summaries not shown in the standard PROC MEANS output. These can be controlled by the WAYS statement; however, we will subset them during the transformation process. There are four variables containing the MPG quartiles, but SGPANEL graph uses one in the role of the RESPONSE= variable. The two different quartiles are values of a panel variable, and the city and highway categories are the group variable in the charts. Clearly, some transformation is required.

Output 1: Data Set from PROC MEANS OUTPUT Statement

Obs	Origin	Type	_TYPE_	_FREQ_	City_Q1	Highway_Q1	City_Q3	Highway_Q3
1			0	254	17	24	21	29
2		SUV	1	35	14	18	16	21
3		Sedan	1	168	18	25	21	30
4		Sports	1	32	16	23	19	27
5		Wagon	1	19	18	25	22	30
6	Europe		2	123	17	24	20	29
7	USA		2	131	17	24	21	30
8	Europe	SUV	3	10	13	16	16	21
9	Europe	Sedan	3	78	18	25	20	29
10	Europe	Sports	3	23	16	23	20	28
11	Europe	Wagon	3	12	18	25	21	29
12	USA	SUV	3	25	14	18	16	21
13	USA	Sedan	3	90	18	26	22	32
14	USA	Sports	3	9	17	24	18	25
15	USA	Wagon	3	7	19	26	26	34

Transforming the Data Set for Use in the Panel Graph

First, Program 2 uses PROC TRANSPOSE to create a data set with a single column of MPG quartiles.

Program 2: Transposing the Quartile Data

```
proc transpose data=summary
    out=summary2(drop = _type_ _label_❶ rename=(col1=MPG)❷);
  where not missing(origin);❸
  by _type_ origin type;❹
  var city: highway:;
run;

proc print data=summary2;
run;
```

❶ These variables of no use after this step, but the DROP= option must be here for both: _type_ is used in the BY statement, _label_ is created by PROC TRANSPOSE.

❷ This renaming is certainly not required, but provides a more intuitive name to the new column of MPG quartiles.

❸ Here, we remove the cases not needed for the graph.

❹ _type_ is the primary sorting variable, with the secondary and tertiary variables corresponding to the CLASS statement in Program 1.

Output 2: Transposed Quartile Data, Partial Listing

Obs	Origin	Type ❶	_NAME_ ❷	MPG
1	Europe		City_Q1	17
2	Europe		City_Q3	20
3	Europe		Highway_Q1	24
4	Europe		Highway_Q3	29
5	USA		City_Q1	17
6	USA		City_Q3	21
7	USA		Highway_Q1	24
8	USA		Highway_Q3	30
9	Europe	SUV	City_Q1	13
10	Europe	SUV	City_Q3	16
11	Europe	SUV	Highway_Q1	16
12	Europe	SUV	Highway_Q3	21

❶ Values for the missing cases will be filled in to make an intuitive value for use in the chart.

❷ These need to be split into two components across the underscore. Generally, we would not advocate including an underscore in variable names; however, this is a useful exception.

Program 3 finishes the transformation of the data; however, it does contain one fairly inconsequential logic error. Can you find it? The SAS log that follows may be helpful.

Program 3: Final Transformation of the Quartile Data

```
data summary2;
  set summary2;
  if missing(type) then type = 'All';
  Statistic = scan(_name_,2,'_');
  Category = scan(_name_,1,'_');
  drop _name_;
  length statistic category $8;
run;
```

Log 3: Final Transformation of the Quartile Data

```
174         data summary2;
175           set summary2;
176           if missing(type) then type = 'All';
177           Statistic = scan(_name_,2,'_');
178           Category = scan(_name_,1,'_');
179           drop _name_;
180           length statistic category $8;
WARNING: Length of character variable Statistic has already been set.
         Use the LENGTH statement as the very first statement in the
         DATA STEP to declare the length of a character variable.
WARNING: Length of character variable Category has already been set.
         Use the LENGTH statement as the very first statement in the
         DATA STEP to declare the length of a character variable.
181         run;

NOTE: There were 40 observations read from the data set WORK.SUMMARY2.
NOTE: The data set WORK.SUMMARY2 has 40 observations and 5 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

Making the Panel Graph

Now the data is in a form that will work well with PROC SGPANEL. Program 4 creates the same chart shown in Graph 1.

Program 4: Making the Panel Graph

```
proc sgpanel data=summary2;
  panelby origin statistic ❶ / layout=lattice novarname ❷;
  vbar type / group=category response=MPG groupdisplay=cluster;❸
  rowaxis label='MPG' offsetmin=0;❹
  colaxis display=(nolabel);❹
  keylegend / position=top title='';❹
run;
```

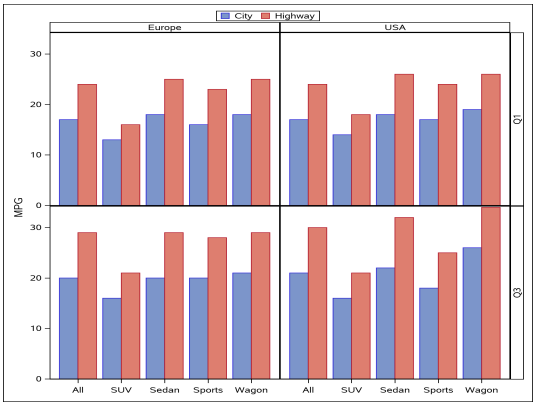
❶ Origin and Statistic are the two variables that define the four panels.

❷ The LATTICE layout puts the first variable on the columns, second on the rows. NOVARNAME suppresses the variable name in the panel labels.

❸ Each panel is a grouped bar chart; charting variable is Type, group variable is Category (City/Highway), and response is MPG.

❹ AXIS and KEYLEGEND statements are used to alter default labels and styles.

Output 4: The Panel Graph



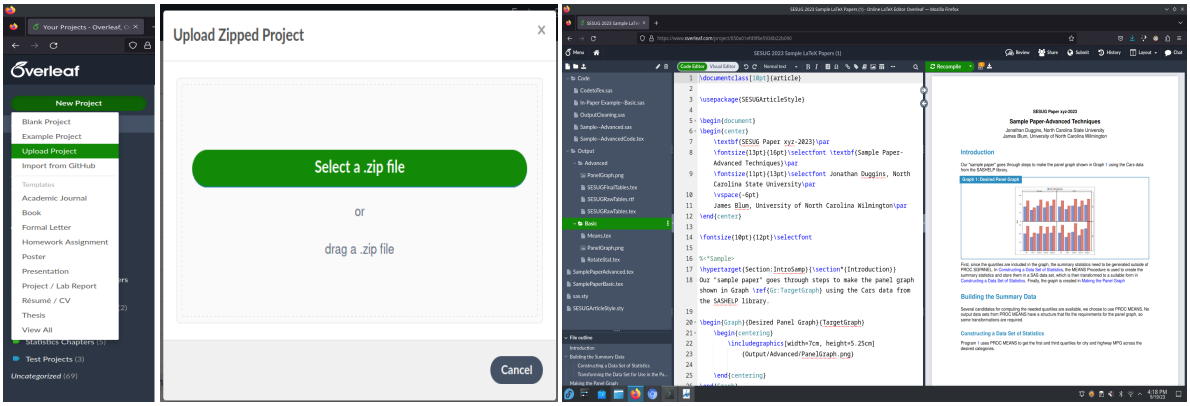
And so ends our sample paper.

Return to the Full Paper Introduction

Appendix B–Downloading the Sample Paper and Setting Up Overleaf

When you download the sample file from Demo Project File, you are downloading a .zip file. Overleaf is equipped to create an entire project from an appropriate .zip file. First, select "Upload Project" from the drop-down menu given by the "New Project" button, as shown in the first panel of Figure 2. This pops up a box to select a .zip file from your computer (second panel)–navigate to the file you downloaded and select OK, or drag your file to the box. If you are successful, Overleaf will immediately open the project, and you will see an environment like the third panel of Figure 2, which is also like the one shown in Figure 1.

Figure 2: Uploading a Project to Overleaf



Appendix C–Extra Code Details

This paper is surely not intended to be a comprehensive tutorial on LaTeX programming; however, we do want to give some additional information if you are interested in tweaking or defining your own LaTeX elements.

Latex Code 12 displays the options used to set general formatting for the sample paper to match the SESUG template.

LaTeX Code 12: General Formatting Options

```
\usepackage{helvet}❶
\renewcommand{\familydefault}{\sfdefault}❶

\usepackage{ragged2e}
\raggedright
\raggedbottom❷

\usepackage[letterpaper,left=1in,right=1in,top=0.99in,bottom=0.75in]
    {geometry}❸

\usepackage{parskip}❹
```

❶ Helvet loads the Helvetica fonts. The RENEWCOMMAND here is applied to the default font family setting it to the sans font default (which is now Helvetica).

❷ By default, LaTeX uses simultaneous left and right justification horizontally, along with top and bottom justification vertically. To prevent this stretching, the Ragged2e package is available: the RAGGEDRIGHT command sets up left justification only, RAGGEDBOTTOM sets top justification only.

❸ The Geometry package sets up page geometry. Optional arguments include paper size and margins.

❹ The Parskip package suppresses indentation at the beginning of paragraphs.

LaTeX Code 13 displays the options used for the LstListing environment in the sample paper.

LaTeX Code 13: Options for the LstListing Environment

```
\usepackage{listings,lstautogobble}❶
\usepackage{courier}❷
\lstset{autogobble=true,❶
    showstringspaces=false,❸
    basicstyle=\ttfamily,❷
    upquote=true❹}
```

❶ The Listings package includes LstListing, the LstAutoGobble package allows for the AutoGobble option in LSTSET. AutoGobble is designed to remove extra indentation used in your LaTeX code within an LstListing.

❷ The Courier font is loaded and used in the LSTSET option BasicStyle (TTFAMILY is the default true type font).

❸ ShowStringSpaces has a default of true, which causes spaces inside literals to be annotated with an underscore. We advocate turning this off.

❹ Single quotes are displayed as upticks by default, UpQuote = True changes to a standard quote character.

Latex Code 14 shows how the CallOut function is defined.

LaTeX Code 14: The CallOut Function Definition

```
\usepackage{pifont}❶

\newcommand{\CallOut}❷[2][prgHeadBlue]❸
    {\textcolor{#1}{\ding{\the\numexpr #2+201}}}❹
```

❶ The PiFont package includes the Wingdings font.

❷ The NEWCOMMAND command first names the command to be created.

❸ Next, in square brackets, the number of arguments of the command/function is given. This is followed by default values for any optional arguments (also in square brackets).

❹ The final argument to NEWCOMMAND is the action taken by the command, detailed below.

The CallOut command sets the text color to the first, optional parameter value when given, or the default of prgHeadBlue if not. That text color is applied to a chosen character from the Wingdings font (DING). The argument to DING is the ASCII value for the character. Since the symbol ❶ has the ASCII value 202, ❷ is 203, and so forth; we choose the right call out character by getting the value (THE) of the sum of the required parameter and a base value of 201 (NUMEXPR).