

SAS® GLOBAL FORUM 2021

Paper 1134-2021

Harnessing Raw Data from Submission Forms to Gauge Food Insecurity Among College Students

Zoraya Cruz-Bonilla, Binghamton University

ABSTRACT

Submission forms are a common practice for collecting student information quickly and directly. The pandemic outbreak has spurred this practice even further in order to pivot in-person campus operations to a safer environment or platform. However, not all submission forms are created equal. Some lack adequate validation rules to impose data integrity; while others fall short on their promise to provide greater meaning due to their flat architecture. Thus, valuable insights easily get lost among the hundreds of records generated. Through the integration of SAS Enterprise Guide and SAS Visual Analytics, participants will learn a handful of functions to consolidate raw data from a series of weekly CSV files; they will learn how to create a handful of custom fields to underscore service usage trends; and how to create on-demand reporting. The outcome is a transformative process that renders raw data to information that is actionable and supports public service efforts.

INTRODUCTION

Since the onset of the COVID-19 pandemic, efforts to mitigate infection rates such as social distancing have limited traditional mechanisms of data collection. In lieu of in-person interactions, there has been a push towards using online submission forms to collect data directly from primary sources. For the Binghamton University Food Pantry, this alternate format not only allowed it to continually track service usage. With the assistance of Student Affairs Assessment, the data was transformed to meaning metrics that stakeholders can respond to.

The purpose of this paper is to provide some supplementary background information and outline a few of the analytic operations that Student Affairs Assessment performed to glean meaningful metrics from submission form data. Which, as you may already know, can be messy, might require a fair amount of cleaning, or must be manipulated in some way to make it more useable for analysis. Specifically speaking, the paper will:

- Showcase the aggregation of raw data from several *.csv files;
- Demonstrate functions for parsing time-series data;
- Introduce options for embedding program code in a point-and-click interface; and
- Briefly highlight the integration of SAS Visual Analytics for data exploration and reporting

As you read through this paper, you'll notice that I use a combination of point-and-click methods and program code. As a relatively notice SAS® Enterprise Guide® user, I find that starting with the former has expanded my understanding of what the analytic tool can do. Yet, I recognize that the point-and-click environment is only scratching the surface. There are hundreds of online community pages that have inspired me to incorporate program code within the same process flow. It has been quite a journey in understanding and applying

what I have learned through these resources, but the gains have taken my skills to the next level. Therefore, I encourage you to take the leap as well.

Now, let's get down to business.

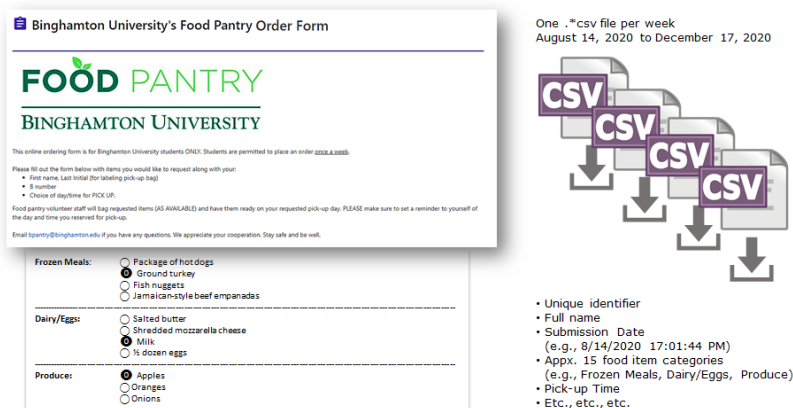
TECHNIQUES FOR WRANGLING WITH DATA FROM MULTIPLE SUBMISSION FORMS

A submission form is a data-gathering tool that allows you to collect different pieces of information from respondents. While choosing the best submission form may depend on available institutional resources, there are several key features that should be considered. Ideally, the tool of choice should have data validation controls because the raw data will be in better shape for analysis. But if it is not possible to acquire a tool with data validation controls, then the question or data input prompts must account for differences in the way that people enter values (e.g., 05/17/2021 versus March 17, 2021).

Towards the end of Spring 2020, the Binghamton University Food Pantry began to collect service user data via a platform that is well integrated into the campus; several offices have successfully utilized the platform for their workflow management processes. And unlike other options that were previously utilized, the platform authenticates user access, auto-populates key identifiers, and prevents more than one submission form per student per week. The following techniques were used to analyze submission form data collected during the Fall 2020 academic semester (August 14, 2020 - December 17, 2020).

As illustrated in Display 1, the online submission form contained several sections. While some sections were purely instructive in nature, others were intended to gather basic information about service impact, needs, and eligibility. Such as, "Will this order provide assistance to a child in your household?" or, "I agree that the Food Pantry is only for enrolled Binghamton University students facing hard times." Particularly important for gauging food insecurity among college students, however, was the last section – where the student must specify food categories.

Each week the form opened for new submissions on Friday at 5 PM and closed the following Friday at 10:00 AM. Every time the form was closed, a new comma-delimited (*.csv) file was generated and saved to a shared drive for the next step in SAS® Enterprise Guide® (version 7.15).



The image shows a screenshot of the Binghamton University's Food Pantry Order Form on the left and a diagram on the right illustrating the weekly export process. The form includes sections for Frozen Meals, Dairy/Eggs, and Produce, with radio button options for each. The diagram shows a stack of CSV files with arrows pointing down, indicating a weekly export cycle from August 14, 2020, to December 17, 2020.

One *.csv file per week
August 14, 2020 to December 17, 2020

- Unique identifier
- Full name
- Submission Date (e.g., 8/14/2020 17:01:44 PM)
- Appx. 15 food item categories (e.g., Frozen Meals, Dairy/Eggs, Produce)
- Pick-up Time
- Etc., etc., etc.

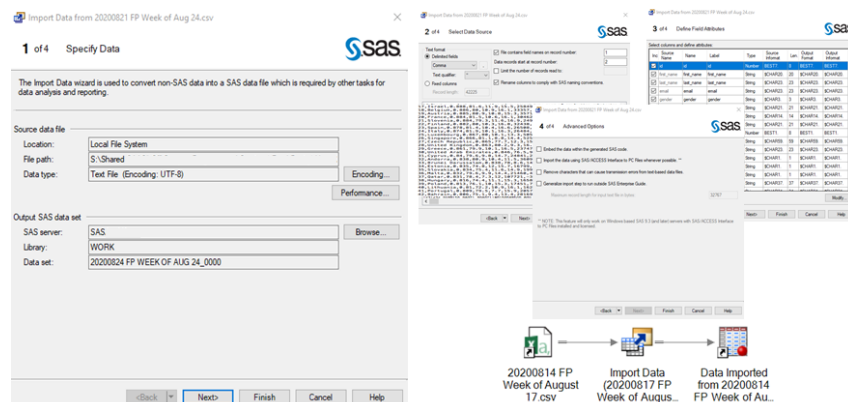
Display 1. Submission Forms are Exported on a Weekly Basis

Overall, 21 *.csv files were imported to a SAS® Enterprise Guide® process flow. Importing *.csv files only takes a few steps due to the easy point-and-click graphical interface:

1. Select *File*
2. Select *Import Data*
3. Select the file location (e.g., *My Computer*)
4. Navigate to the file folder
5. Select the *.csv file
6. Click *Open*

Depending on how many *.csv files you are working with, however, consider whether using the IMPORT procedure is more time-efficient. Check out the online SAS® community pages to explore how to import multiple *.csv files using only a few lines of code.

Opening the *.csv file will invoke the import wizard which allows you to specify text qualifiers (e.g., commas), the row number containing field names, and the option to limit the number of records for SAS® to read. This is helpful for files that do not follow a conventional or expected layout. Fortunately, Step 3 of the wizard [Define Field Attributes] determines how the data fields should be formatted in the output – or if they even need to be in the output at all! Also, string fields should have a length value that is as many characters as that of the longest string in the corresponding column. Otherwise, a value for *Frozen Meals* will display as “*Chicken patties, Chicken meatballs*” instead of, “*Chicken patties, Chicken meatballs (gluten-free soy-free).*” If qualitative data analysis is not applicable, length is irrelevant.



Display 2. Select Columns and Define Attributes Through the Import Wizard

AGGREGATING RAW DATA

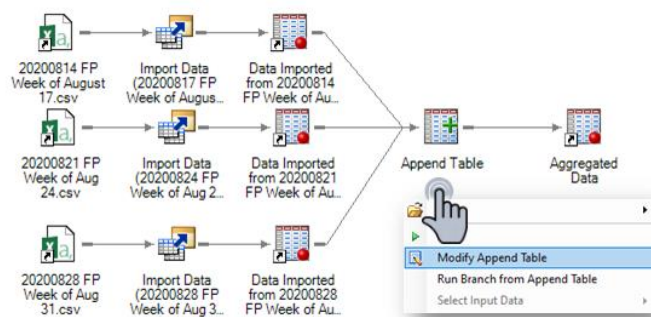
After importing each *.csv file, individual tables are created in the process flow. Ideally, submission forms should be in one comprehensive data set. This makes it easier to apply data manipulation techniques. The online SAS® community pages contain several methods for aggregating data. In this paper, I will show two methods and the rationale for using them at specific stages of the data analysis.

METHOD 1: Using the APPEND TABLE Task is as Easy as Pie!

The APPEND TABLE task is perhaps the most user-friendly and straightforward method for novice SAS® users to aggregate data. The following shows how to invoke the APPEND TABLE task directly from the process flow:

1. Select *Tasks*
2. Select *Data*
3. Select *Append Table...*
4. Select *Add Table...*
5. Select the tables to append (e.g., *Data Imported from 20200814 FP Week of August 17.csv*)
6. Ensure all applicable tables are listed in the dialogue box
7. Click *Run*

The steps will render join lines between the input and output tables. Also, right-clicking on the new node in between the tables brings up the option to modify the APPEND TABLE task (see Display 3).



Display 3. The APPEND TABLE task creates joins between input and output tables

Gotcha!

One of the learned lessons from using the APPEND TABLE task is that the column header names determine how the data is displayed in the output table. To illustrate, let's consider what happens when using column header names shown on Display 4.

The data is for weeks 1, 2, and 3. Throughout multiple weeks, the column header names pertaining to the user's ID number, full name, submission date, and the food category for frozen meat remain unchanged. The food category for fresh fruit and vegetables, however, did change due to product availability.

WEEK 1	id	full_name	submission_date	FROZEN Meat (Pick 2 Total)	Fresh fruit and vegetables (Limit 2 choices total per person)
	123	---	---	---	---
	345	---	---	---	---
WEEK 2	id	full_name	submission_date	FROZEN Meat (Pick 2 Total)	Fresh Produce (choose 1 per person)
	123	---	---	---	---
	678	---	---	---	---
WEEK 3	id	full_name	submission_date	FROZEN Meat (Pick 2 Total)	Fresh Produce (choose 2 per person)
	123	---	---	---	---
	345	---	---	---	---

Display 4. Food Categories with Different Column Header Names on Subsequent Weeks

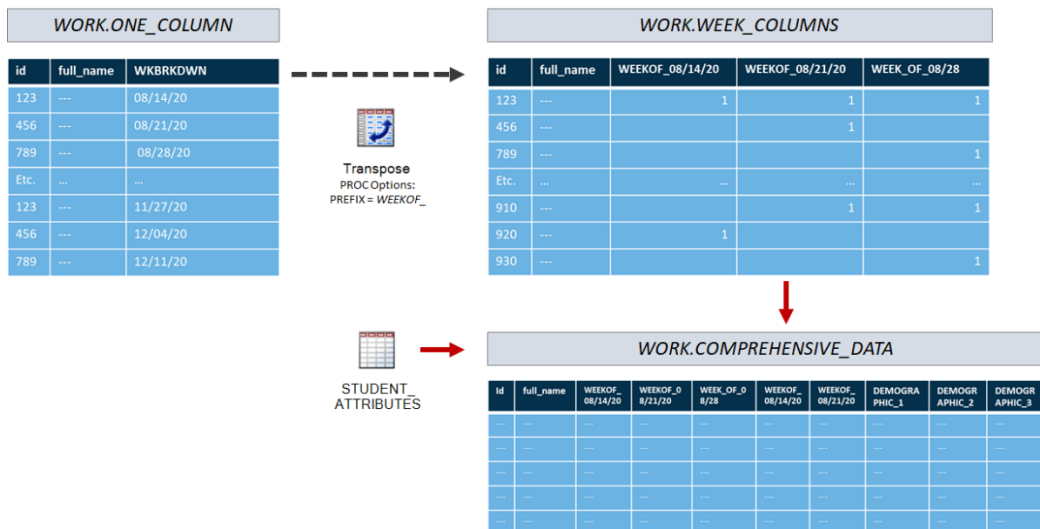
The APPEND TABLE task successfully reads all these column header names but will interpret these differences as distinct variables. Display 5 is an exemplary rendition of how SAS® will display the data in the output table. Notice the arrangement of the raw data which expands both vertically (to account for each submission form), and horizontally (to account for any possible food category name changes). This is convenient if you need to analyze the food categories separately but leads to additional data manipulation procedures if you need to, for example, compare the selection of vegetable versus fruit throughout the entire academic semester. In that case, the CATX function will be your new (non-furry) best friend! More about functions in the next section.

id	full_name	submission_date	FROZEN Meat (Pick 2 Total)	Fresh fruit and vegetables (Limit 2 choices total per person)	Fresh Produce (choose 1 per person)	Fresh Produce (choose 2 per person)
123	---	9/25/2020 17:03	Chicken meatballs (15) fully cooked (gluten-free soy-free), Breaded chicken strips (10), Chicken patties (3)	Orange, Apple, Baby carrots (baggie), Onion, Potato, Head of lettuce (while supplies last), Other (Feel free to comment below)		
345	---	9/29/2020 18:21	Falafel fritters (gluten-free, vegan) - uncooked	Baby carrots (baggie), Onion (2), Potato		
123	---	10/2/2020 0:16			Potato	
678	---	10/4/2020 19:01	Breaded chicken strips (10), Fish filet (2) - white fish		Potato	
123	---	10/19/2020 3:27	Chicken meatballs (15) fully cooked (gluten-free soy-free), Breaded chicken strips (10)			Apple (while supplies last), Orange/ clementine (while supplies last)
345	---	10/20/2020 12:29				Potato, Onion

Display 5. Food Categories with Different Column Header Names Displayed Horizontally

METHOD 2: Using the MATCH-MERGE Program Code to Mix Things Up

Another important takeaway about the APPEND TABLE task output is that the rows are independent of each other. In other words, the output is an accumulation of observations – as opposed to a condensed version joined by a common identifier. While this is not problematic for an initial data exploration, carrying out additional types of tasks in the same process flow results in several data subsets that must then get merged to form a complete picture. For example, transposing variables to capture a list of unique food pantry users (see Display 6) and importing secondary sources are two output tables from two distinct tasks.



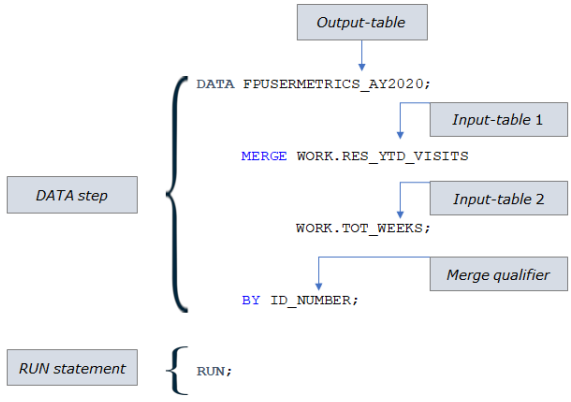
Display 6. Transposing Variables Creates One Row per Student BUT is a Separate Data Subset from Secondary Sources in the SAS® Process Flow

Using program code is a good choice for matching and merging disparate data sets. This is the program code used to match-merge records:

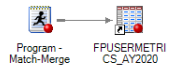
```
DATA FPUSERMETRICS_AY2020;
  MERGE WORK.RES_YTD_VISITS WORK.TOT_WEEKS;
  BY ID_NUMBER;
RUN;
```

Let's dissect this to understand the fundamental components. The first part is to assign a name for the output table in the DATA step. Next, the MERGE statement must be called upon to instruct the program what to do with the two input tables. The BY statement controls the operation of the MERGE statement and sets up special grouping variables. The BY statement also specifies the column that will have row values sorted. By default, SAS sorts in ascending order. Finally, all DATA steps must end with a RUN statement and a semicolon. As illustrated in Display 7, the lines of code must be entered onto the program editor. Once processed successfully, the output table will be rendered in the process flow.

Program Editor



Process Flow



Display 7. Program Code in the DATA Step to MATCH-MERGE Records

FUNCTIONS IN A COMPUTED COLUMN

Computed columns offer the flexibility of performing calculations or change values for the purposes of broadening the analytic power of existing data. The imported field, 'SUBMISSION_DATE,' has an icon next to the name which denotes that it is comprised of character values (see Display 8). For the purposes of using this field to determine the day of the week for each submission form, a computed column must be created to transform the field to a DATETIME value.

Imported Fields				Computed Columns				
SUBMISSION_DATE	BREAD_CER EAL_DESC	CAN_BEANS TEV1DESC	DRY PASTA RICE_DESC	FROZE MEA L_DESC	DATETIME	WEEKDAY	WEEKNOF	WEEKR4ZIN
9/30/2020 12:26	Naan bread (f)	Garbonzo bean.	Tri-colored ou.	Falafel fritters (.	9/30/2020 12:26 PM	Wednesday	09/28/20	09/25/20
9/30/2020 13:32	Box of cereal (.	Black beans	Spaghetti.Brow.	Breaded chick.	9/30/2020 1:32 PM	Wednesday	09/28/20	09/25/20
9/30/2020 14:10	Naan bread (f)	Garbonzo bean.	Tri-colored ou.	Falafel fritters (.	9/30/2020 2:10 PM	Wednesday	09/28/20	09/25/20
9/30/2020 16:05	Naan bread (f)	Garbonzo bean.	Tri-colored ou.	Breaded chick.	9/30/2020 4:05 PM	Wednesday	09/28/20	09/25/20
9/30/2020 16:50	Naan bread (f)	Garbonzo bean.	Tri-colored ou.	Breaded chick.	9/30/2020 4:50 PM	Wednesday	09/28/20	09/25/20
9/30/2020 17:00	Naan bread (f)	Garbonzo bean.	Tri-colored ou.	Breaded chick.	9/30/2020 5:00 PM	Wednesday	09/28/20	09/25/20
9/30/2020 18:29	Naan bread (f)	Garbonzo bean.	Tri-colored ou.	Breaded chick.	9/30/2020 6:29 PM	Wednesday	09/28/20	09/25/20
9/30/2020 18:40	Box of cereal (.	Black beans	Spaghetti.Brow.	Falafel fritters (.	9/30/2020 6:40 PM	Wednesday	09/28/20	09/25/20
9/30/2020 20:34	Naan bread (f)	Garbonzo bean.	Tri-colored ou.	Breaded chick.	9/30/2020 8:34 PM	Wednesday	09/28/20	09/25/20
9/30/2020 21:57	Naan bread (f)	Garbonzo bean.	Tri-colored ou.	Falafel fritters (.	9/30/2020 9:57 PM	Wednesday	09/28/20	09/25/20
9/30/2020 22:26	Naan bread (f)	Dark red kidne.	Tri-colored ou.	Falafel fritters (.	9/30/2020 10:26 PM	Wednesday	09/28/20	09/25/20
10/1/2020 1:08	Naan bread (f)	Dark red kidne.	Whole wheat s.	Chicken pattie.	10/1/2020 1:08 AM	Thursday	09/28/20	09/25/20
10/1/2020 2:14	Naan bread (f)	Garbonzo bean.	Tri-colored ou.	Breaded chick.	10/1/2020 2:14 AM	Thursday	09/28/20	09/25/20
10/1/2020 7:25	Naan bread (f)	Garbonzo bean.	Tri-colored ou.	Falafel fritters (.	10/1/2020 7:25 AM	Thursday	09/28/20	09/25/20
10/1/2020 10:51	Naan bread (f)	Garbonzo bean.	Spaghetti.Vhol.	Breaded chick.	10/1/2020 10:51 AM	Thursday	09/28/20	09/25/20
10/1/2020 12:31	Naan bread (f)	Garbonzo bean.	Tri-colored ou.	Breaded chick.	10/1/2020 12:31 PM	Thursday	09/28/20	09/25/20
10/1/2020 12:40	Naan bread (f)	Garbonzo bean.	White rice Bro.	Chicken meatb.	10/1/2020 12:40 PM	Thursday	09/28/20	09/25/20
10/1/2020 12:46	Package of 12.	Dark red kidne.	Spaghetti.Brow.	Chicken meatb.	10/1/2020 12:46 PM	Thursday	09/28/20	09/25/20
10/1/2020 13:10	Naan bread (f)	Dark red kidne.	Whole wheat s.	Falafel fritters (.	10/1/2020 1:10 PM	Thursday	09/28/20	09/25/20
10/1/2020 14:04	Naan bread (f)	Garbonzo bean.	Tri-colored ou.	Chicken pattie.	10/1/2020 2:04 PM	Thursday	09/28/20	09/25/20
10/1/2020 17:37	Box of cereal (.	Black beans	Tri-colored ou.	Breaded chick.	10/1/2020 5:37 PM	Thursday	09/28/20	09/25/20
10/1/2020 18:47	Naan bread (f)	Garbonzo bean.	Tri-colored ou.	Breaded chick.	10/1/2020 6:47 PM	Thursday	09/28/20	09/25/20
10/1/2020 19:13	Naan bread (f)	Garbonzo bean.	Tri-colored ou.	Breaded chick.	10/1/2020 7:13 PM	Thursday	09/28/20	09/25/20
10/1/2020 20:23	Naan bread (f)	Garbonzo bean.	Spaghetti.Vhol.	Falafel fritters (.	10/1/2020 8:23 PM	Thursday	09/28/20	09/25/20
10/1/2020 23:04	Package of 12.	Garbonzo bean.	Tri-colored ou.	Breaded chick.	10/1/2020 11:04 PM	Thursday	09/28/20	09/25/20
10/1/2020 23:26	Naan bread (f)	Garbonzo bean.	Tri-colored ou.	Falafel fritters (.	10/1/2020 11:26 PM	Thursday	09/28/20	09/25/20
10/2/2020 0:16	Naan bread (f)	Dark red kidne.	Spaghetti.Vhol.	Falafel fritters (.	10/2/2020 12:16 AM	Friday	09/28/20	09/25/20
10/2/2020 0:44	Package of 12.	Dark red kidne.	Tri-colored ou.	Falafel fritters (.	10/2/2020 12:44 AM	Friday	09/28/20	09/25/20
10/2/2020 17:00	Oatmeal packe.	White rice	Fish filet (2) - w.	White rice	10/2/2020 5:00 PM	Friday	09/28/20	10/02/20
10/2/2020 17:03	Naan bread (f)	Black beans	Brown rice	Breaded chick.	10/2/2020 5:03 PM	Friday	09/28/20	10/02/20
10/2/2020 19:38	Naan bread (f)	Garbonzo bean.	Whole wheat s.	Chicken meatb.	10/2/2020 7:38 PM	Friday	09/28/20	10/02/20
10/2/2020 20:13	Oatmeal packe.	White rice	Frozen mac &.	Frozen mac &.	10/2/2020 8:13 PM	Friday	09/28/20	10/02/20
10/2/2020 21:12	Naan bread (f)	Black beans	Whole wheat s.	Chicken meatb.	10/2/2020 9:12 PM	Friday	09/28/20	10/02/20

Display 8. Transforming Character Values to DATETIME Values via a Function in a Computed Column

Here are the steps to create a computed column:

1. Computed Columns
2. New
3. Advanced Expression
4. Next

The expression used to evaluate the day of the week is, `COMPRESS((PUT(((DATEPART(t1.DATETIME))), WEEKDATE9.)), 's')`.

Essentially, the expression is extracting the date part from DATETIME values, changing it a character format, and uses a modifier to remove spaces from the string. The output are values such as Monday, Tuesday, Wednesday, etc.

Knowing the best function to use in a computed column is paramount for making the most use out of the SAS® environment. And with well over 100 pre-defined functions to choose from, it is bound to supercharge complex processes. It is certainly worth the time to explore documentation pages for additional functions or to “Frankenstein” new applications.

WRITING PROGRAM CODE TO CALCULATE FREQUENCY OF VISITS

Earlier in this paper, writing program code was demonstrated through a simple MATCH-MERGE data step. In the current section, program code will be used to quantify the total number of weeks that each student visited the Binghamton University Food Pantry during the Fall 2020 academic semester. Therefore, rather than using the SUM function in a computed column, a few lines of code will be written. But before diving into the statement components, let’s examine *why* it is a more favorable approach.

A careful review of the TRANSPOSE function previously shown on Display 6 reveals that weeks are spread across multiple columns – a necessary step to collate the data as one row per student. Also, column header names are a concatenation of WKBRKDOWN values and the ‘WEEKOF_’ prefix to make the name more descriptive. If the student visited the Food Pantry, then it is coded as ‘1’ for the corresponding column. Using the SUM function in a computed column to add up all the values of ‘1’ across the row requires manual entries of every column header name onto the expression. This is not convenient for two reasons. Foremost, it is a bit time-consuming to enter every column header name. Most importantly, however, manual entries are error prone. As data analysts, we want to automate as much as possible. Writing a few lines of code eliminates the need to revise the expression week after week.

WORK.WEEK_COLUMNS				
id	full_name	WEEKOF_08/14/20	WEEKOF_08/21/20	WEEK_OF_08/28
123	---	1	1	1
456	---		1	
789	---			1
Etc.
910	---		1	1
920	---	1		
930	---			1

Display 9. Tabulation of Week Columns

This is the program code used to quantify the total food pantry visits per student:

```
DATA TOT_WEEKS;
  SET WORK.WEEK_COLUMNS;
  TOT_WEEKS=SUM(of WEEKOF_);
  PUT_ALL_;
RUN;
```

Line 3 of the code instructs the program to sum up all the values that have a variable beginning with the prefix, 'WEEKOF_' (collectively referred to as a *variable-list*). Clearly, it is a much shorter alternative.

WRITING PROGRAM CODE TO EXTRAPOLATE STUDENTS' FIRST FOOD PANTRY SERVICE REQUEST

Numerous news outlets made it clear that the onset of the COVID-19 pandemic exacerbated food insecurity. This was certainly also the case among college students. According to survey data from a large multi-institutional sample (n = 38,602 students in 26 states), food insecurity affected 38% of students enrolled in a four-year institution and 44% of students enrolled in a two-year institution (Goldrick-Rab et al., 2020). While the survey was administered in the early days of the pandemic, it is very likely that those percentages went up due to worsening economic conditions coupled with unprecedented challenges faced by students that did not need food pantry services prior to March 2020. Therefore, tracking service usage patterns became an important piece of the puzzle towards gauging unmet needs among college students. The implementation of an online submission form – which was initially intended to swap in-person visits with a safer alternative to food access for Binghamton University students – allowed the means to analyze the data from the standpoint of how usage patterns have changed in response to the pandemic and contributed to our expanding understanding of students in crisis.

The program code to extrapolate students' first Food Pantry service request is:

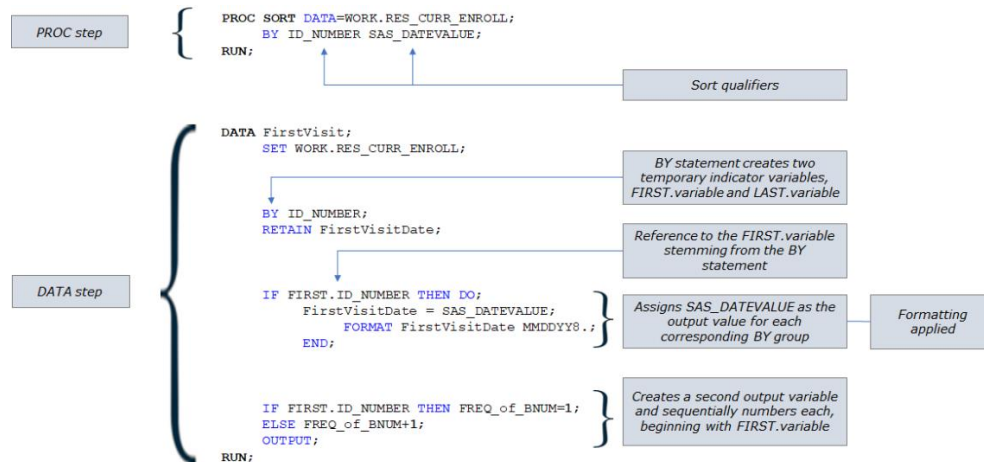
```
PROC SORT DATA=WORK.RES_CURR_ENROLL;
  BY ID_NUMBER SAS_DATEVALUE;
RUN;
DATA FirstVisit;
  SET WORK.RES_CURR_ENROLL;
  BY ID_NUMBER;
  RETAIN FirstVisitDate;
  IF FIRST.ID_NUMBER THEN DO;
    FirstVisitDate = SAS_DATEVALUE;
    FORMAT FirstVisitDate MMDDYY8.;
  END;
  IF FRIST.ID_NUMBER THEN FREQ_of_BNUM=1;
  ELSE FREQ_of_BNUM+1;
  OUTPUT;
RUN;
```

It is slightly more complex than the lines of code we have encountered in earlier sections of this paper. Breaking down the statement components should help.

The program begins with a PROC step to sort the input table – first by ID number and then by submission date. Next, the DATA step assigns the output table as 'FirstVisit.' The BY statement automatically creates two temporary indicator variables, FIRST.variable and LAST.variable. A new column is created, 'FirstVisitDate,' to capture the information of interest in the output table. If the observation is one of the temporary indicator variables

corresponding to the First.variable (i.e., 'First.ID_Number'), then it will be equal to the submission date and formatted to display as a two-digit month, two-digit day, and two-digit year. It will also sequentially number each observation stemming from the same BY group in another column, 'FREQ_of_BNUM.' If the observation is not one of the temporary indicator variables, then it will display the same date value until it encounters a new BY group (i.e., different ID number).

To learn more about how SAS® determines the FIRST.variable and LAST.variable, go to relevant documentation pages.



Display 10. Program Code to Identify Students' First Food Pantry Service Request

A rendition of the results for the program code is shown on Display 11.

id	full_name	SUBMISSION_DATE	FirstVisitDate	FREQ_of_BNUM
123	---	09/04/20	09/04/20	1
123	---	09/11/20	09/04/20	2
123	---	09/25/20	09/04/20	3
123	---	10/02/20	09/04/20	4
456	---	09/25/20	09/25/20	1
456	---	10/02/20	09/25/20	2

Display 11. Tabulation of 'FirstVisitDate'

INTEGRATING DATA VISUALIZATION

Data is great. Visualizations are even better. To be more specific, a *good* visualization is better. One of the features of SAS Visual Analytics is that it is intuitive enough to automatically make chart suggestions. Therefore, expediting reporting capabilities. But, of course, it's also possible to create charts or graphs from scratch and to customize pages according to specific data needs, and comply with intuitional branding.

With the use of SAS Visual Analytics, Food Pantry report metrics contained:

- The total service requests per month;
- The number of new users for each subsequent month of the semester;
- A comparison of students living on-campus versus off-campus;
- A breakdown of graduate versus undergraduate students;
- A comparison between the proportion of users based on residency versus the proportion observed in the entire student population for the given semester;
- Food pantry users who are first-generation students or participate in the Educational Opportunity Program (EOP);
- And more

It is worth noting that SAS Visual Analytics has interactive capabilities; it can be used to share the information dynamically by adding control buttons. These buttons enable administrators to drill down on the data and offer the flexibility for on-demand reporting.

CONCLUSION

The data collected through online submission forms had broader uses. With the assistance of Student Affairs Assessment, the Binghamton University Food Pantry was able to launch a survey in late December to determine how the onset of the pandemic affected users and an infographic was created to compare usage between Fall 2019 and Fall 2020. In addition, the data has provided the necessary information for grant proposals.

REFERENCES

Cody, R. (2010). *SAS® Functions by Example, Second Edition*. Cary, NC: SAS Institute Inc.

Foley, M.J. (2009). *MERGING vs. JOINING: Comparing the DATA Step in SQL*, Chapel Hill, NC. Available at support.sas.com/resources/papers/proceedings09/036-2009.pdf.

Goldrick-Rab, S., Coca, V., Kienzl, G., Welton, C.R., Dahl, S., & Magnelia, S. (2020). #RealCollege During the Pandemic: New Evidence on Basic Needs Insecurity and Student Well-being. Available at https://hope4college.com/wp-content/uploads/2020/10/Hopecenter_RealCollegeDuringthePandemic_Reupload.pdf

Wicklin, R. (2018). "How to use FIRST.variable and LAST.variable in a BY-group analysis in SAS." Accessed February 26, 2021. blogs.sas.com/content/iml/2018/02/26/how-to-use-first-variable-and-last-variable-in-a-by-group-analysis-in-sas.html.

ACKNOWLEDGMENTS

The author would like to thank Chris Battiston for this mentorship and support.

RECOMMENDED READING

- Stebleton, M. (2020). "Food Insecurity, COVID-19, and Role of Student Affairs Educators." *Journal of College and Character Connexions*, Vol.6, No.3. Available at naspa.org/blog/food-insecurity-covid-19-and-role-of-student-affairs-educators.
- Center for Disease Control. (n.d.). *Food and Food System Resources During COVID-19 Pandemic*. Accessed April 9, 2021. [cdc.gov/nutrition/resources-publications/food-and-food-system-resources-during-covid-19-pandemic.html](https://www.cdc.gov/nutrition/resources-publications/food-and-food-system-resources-during-covid-19-pandemic.html).

CONTACT INFORMATION

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

Zoraya Cruz-Bonilla
Binghamton University
zcruzbon@binghamton.edu

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

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