PharmaSUG 2025 - Paper AP-238

When PROC SORT Isn't Enough: Implementing Horizontal Sorting in SAS®

Krutika Parvatikar, Merck & Co., Inc., Rahway, NJ, USA
Jeff Xia, Merck & Co., Inc., Rahway, NJ, USA

ABSTRACT

Sorting data is a fundamental task in SAS®, typically handled using "PROC SORT". However, certain scenarios require sorting values within an individual observation rather than across rows. This paper explores horizontal sorting techniques in SAS, including array-based sorting and bubble sorting, to address such challenges. A key application in clinical research involves automating the assembly of citations for Tables, Listings, and Figures (TLFs), linking ADaM and SDTM datasets for streamlined review and improved data traceability. By demonstrating innovative sorting approaches, this paper highlights how tailored techniques can enhance efficiency and expand SAS's capabilities in handling complex data scenarios.

INTRODUCTION

Sorting is a crucial operation in data management and analysis, ensuring that information is structured in a meaningful way for efficient retrieval and interpretation. In SAS, sorting is typically performed using the PROC SORT procedure, which is widely used to organize datasets by specified key variables. However, certain data scenarios require sorting operations beyond the conventional dataset-level approach. One such scenario arises when sorting values within an individual observation—often referred to as horizontal sorting, where multiple variables need to be ordered within a single row rather than across records.

Since PROC SORT does not natively support horizontal sorting, SAS programmers must implement alternative methods, including arrays, iterative sorting algorithms (e.g., bubble sort), and built-in functions like CALL SORTC for character variables and CALL SORTN for numeric variables. This paper explores these techniques, demonstrating how SAS can be leveraged to handle unique data scenarios that require intra-record sorting. By mastering horizontal sorting methods, programmers can enhance data organization, improve reporting accuracy, and streamline analytical workflows. In contrast to alternative analytical methods, the horizontal data sorting technique offers several notable benefits, including its straightforward nature, user-friendly design, and the generation of easily interpretable analytical outcomes.

HORIZONTAL SORTING USING BUBBLE SORT

Bubble sort is one of the simplest sorting algorithms, often used for educational purposes and small datasets. In SAS, it is particularly useful for horizontal sorting where we primarily sort values within a single observation (row). Since PROC SORT only sorts data vertically (by rows), bubble sort can be implemented within a DATA step using arrays and loops to reorder variables within an observation. It allows programmers to reorder multiple values within a single row using arrays and loops in a DATA step. Though computationally inefficient for large-scale sorting, its simplicity and ease of implementation make it valuable in specific SAS applications.

Bubble sort operates by iteratively comparing and swapping adjacent elements within an array until all values are sorted in the desired order. The algorithm follows these steps:

1. Start with the first element in the array.
2. Compare the current element with the next element.
3. If the two elements are out of order (i.e., in ascending order, the first is greater than the second), swap them.

4. Move to the next pair and repeat the comparison and swapping process.
5. After each complete pass through the array, the largest (or smallest) element moves to its correct position at the end.
6. The process is repeated for the remaining unsorted elements until no swaps are required, indicating that the array is fully sorted.

Since each pass moves the largest (or smallest) element into place, fewer comparisons are needed in subsequent iterations, reducing the number of operations gradually.

```
data sorted_data;
   set clinical_data;
   array names[10] names1-names10;
   flag_swap = 1;

   do i = 1 to dim(names) - 1 while (flag_swap=1);
       flag_swap = 0;
       do j = 1 to dim(names) - i;
           if names[j] > names[j+1] then do;
               temp = names[j];
               names[j] = names[j+1];
               names[j+1] = temp;
               flag_swap = 1;
           end;
       end;
   end;
run;
```

Program 1. Code snippet to demonstrate usage of bubble sort to sort the lab test results values.

HORIZONTAL SORTING USING CALL SORTC

Another method for performing horizontal sorting is the CALL SORTC routine, which was introduced in Version 9.2 of SAS (released in 2008). This routine is part of the SAS® Component Language (SCL) and is specifically designed to sort character data in a defined order. It is particularly useful when working with arrays that contain character strings, as it provides a more efficient sorting process compared to other methods.

The basic syntax for CALL SORTC routine is as follows: CALL SORTC(array, length). In this syntax, “array” refers to the character array you wish to sort. The array should be defined with the appropriate dimensions and lengths, while “length” indicates the length of each string in the array. Specifying this length ensures that the routine understands the maximum character length and sorts the strings accordingly.

For example, after applying the CALL SORTC routine to the array named "domains" in the following code snippet, the elements will be arranged in alphabetical order.

```
data sort_example;
   length domain $2;
   array domains[10] $2 ('DS', 'CM', 'EX', 'LB', 'VS', 'SS', 'QS', 'TR', 'RS', 'SV');
   call sortc(domains, 20);
run;
```

Program 2. Code snippet to demonstrate usage of Call SORTC routine.

USE CASE1: ASSEMBLE TEXT IN ALPHABETICAL ORDER IN SUBMISSION DOCUMENTS

PHUSE provided templates for cSDRG and ADRG to be included in submission packages. These documents include many sections that involve a list of SDTM or ADaM dataset names. An important

aspect of these lists is that dataset and variable names should be in alphabetical order. This requirement facilitates the review process by enabling FDA reviewers to locate information quickly and efficiently.

For example, consider an ADRG regarding variable derivation in ADSL. A macro can be developed to take an unsorted domain list as input and provide a sorted list of names, separated by commas, easily and efficiently.

2. The SDTM datasets (CM, DS, EX, LB, QS, RS, SS, SV, TR, and VS) without data cutoff date applied to derive the Uncut Last Known Alive Date (UCLKADT) and Uncut Death Date (UCDTHDT) with Imputation respectively.
3. The SDTM datasets (CM and PR) are used to derive the Start Date of New Anti-cancer Therapy.

Display 1. Screen Print in Section 4.3 in ADRG

There are several sections in the cSDRG and ADRG that require text to be presented in sorted order, such as Section 3.1 in the cSDRG (Overview), Section 3.5 in the ADRG (Imputation/Derivation Methods), Section 4.3 (Intermediate Datasets), and various entries in Section 5.2 (Analysis Datasets). This macro provides a quick and efficient way to assemble the required text in the desired order (alphabetical order, in this case). Performing this task manually can be tedious, error-prone, and less efficient.

Were any domains planned, but not submitted because no data were collected? Yes

If yes, list domains not submitted:

FF (Family Findings), FE (Family Events), IS (Immunogenicity Specimen Assessments), MS (Microbiology Susceptibility), PX (Pregnancy-Related Events), PY (Pregnancy History and Outcomes), RP (Reproductive System Findings) and SR (Skin Response).

Display 2. Screen Print in Section 3.1 in cSDRG

USE CASE 2: ASSEMBLE TEXT IN CITATION FOR TFL IN CSR

A more complex example is to assemble the citations for Tables, Listings, and Figures (TLFs) for CSR. Below is an example of a typical citation in a TLF included in the CSR. It lists all the required datasets involved in generating the displayed analysis in the table. The datasets could be in ADaM or SDTM format. One requirement for the citation text is that it must follow a specific order to improve review efficiency:

1. Display all ADaM datasets before SDTM datasets.
1. Display ADSL before all ADaM datasets.
2. Display DM before all SDTM datasets.
3. The rest of ADaM or SDTM datasets should be displayed in alphabetical order.
4. The SDTM domain and its corresponding SUPPQUAL appear together.

Trial Number: 9999-001, Site: [redacted], Investigator: [redacted], Country: USA

Subject ID	Discontinuation Date	Relative Day of Discontinuation	Type of Discontinuation	Reason For Discontinuation
Pembrolizumab + Chemotherapy				
Trial Number=[redacted], Site Number=[redacted], Unique Subject ID=[redacted], Sex=F, Age=51 Years				
[redacted]	04SEP2020	113	Study Treatment	Progressive Disease
	29OCT2020	168	Trial	Death
Placebo + Chemotherapy				
Trial Number=[redacted], Site Number=0038, Unique Subject ID=[redacted], Sex=F, Age=61 Years				
[redacted]	11FEB2022	323	Study Treatment	Progressive Disease
	01SEP2022	525	Trial	Death

The discontinuation date for study medication is the date of the participant's last dose in the trial.
The discontinuation date for trial or trial segment is the date of assessment.
[a]Associated with COVID-19
Database Cutoff Date: 15DEC2022.

Source: [P001V01MK9999: adam-adsl; adae] [P001V01MK9999: sdtm-dm; suppdm; ae; suppae; ds; suppds; ex; suppex]

Display 3. Screen print of a Table in CSR with citations in the bottom.

As we can see from the above rules, simply sorting the text in alphabetical order using the CALL SORTC routine does not suffice. To work around this issue, we can add specific sorting factors to the front of each individual element in the array. In the following code snippet, sorting factors will be assigned to each text element based on the desired order. For example, the text "ADSL" will be assigned as category 1, the rest of the ADaM datasets will be assigned as category 2; the text "SDTM" will be assigned as category 5; both DM and SUPPDM will be assigned as category 6, and the rest of the SDTM datasets will be assigned as category 8.

```
data readsource2 ;
    set readsource1;
    array all[*] all1-all100;
    do i = 1 to dim(all);
        if index(all[i], 'Source') > 0 or
           anydigit(all[i]) > 0  or
           index(all[i], 'ISS') > 0 or
           index(all[i], 'ISE') > 0 then all[i] = '9';
        else if index(all[i], 'adam') > 0  or
                index(all[i], 'analysis') > 0  then all[i] = '0'||strip(all[i]);

        /*put ADSL before any other ADaM datasets*/
        else if index(all[i], 'adsl') > 0 then all[i] = '1'||strip(all[i]);
        else if substr(all[i], 1, 2) = 'ad' then all[i] = '2'||strip(all[i]);

        /*put SDTM datasets after any other ADaM datasets*/
        else if index(all[i], 'sdtm') > 0 or
                index(all[i], 'tabulations') > 0 then all[i] = '5'||strip(all[i]);

        /*put DM/SUPPDM before any other SDTM datasets*/
        else if index(all[i], 'dm') > 0 then all[i] = '6'||strip(all[i]);
        else if index(all[i], 'suppdm') >0 then do;
            all[i] = '6'||strip(substr(all[i], 5, 2)||'supp');
        end;

        /*put every other SDTM domain in alphabetical order after DM/SUPPDM */
        else if substr(all[i], 1, 4) = 'supp' then do;
            all[i] = '8'||strip(substr(all[i], 5, 2)||'supp');
        end;
        else if compress(all[i]) > '' then  all[i] = '8'||strip(all[i]);

        /*put everything else in the end*/
        else all[i] = '9'||strip(all[i]);
    end;

    call sortc(of all(*));
```

```
    do i = 1 to dim(all);
        if index(all[i], 'supp') > 0 then do;
            all[i] = substr(all[i], 1, 1)||'supp'||substr(all[i], 2, 2);
        end;
    end;
run;
```

Program 3. Code snippet to demonstrate assigning sorting factors based on citation rules.

CONCLUSION

In conclusion, sorting techniques in SAS, such as bubble sort and the CALL SORTC routine, play a crucial role in managing and presenting data effectively. Bubble sort, while simple, allows for horizontal sorting of small datasets within single observations. The CALL SORTC routine, introduced in SAS Version 9.2, efficiently sorts character arrays, improving information organization.

The practical applications of these sorting methods, particularly in preparing submission documents and citations for TFL in CSRs, underscore their importance. Specific sorting requirements, such as prioritizing ADaM datasets over SDTM datasets and adhering to defined display orders—demonstrate the necessity of tailored sorting strategies beyond simple alphabetical arrangements.

By leveraging these sorting techniques, SAS programmers can enhance workflow efficiency and ensure compliance with regulatory standards, ultimately facilitating a smoother review process.

REFERENCES

Hughes, Troy Martin. 2016. "Sorting a Bajillion Records: Conquering Scalability in a Big Data World". Proceedings of the SAS Global Forum, Las Vegas, NV.
https://support.sas.com/resources/papers/proceedings16/11888-2016.pdf

Louise Hadden, Charu Shankar. 2023. “Put on the SAS® Sorting Hat and Discover Which Sort is Best for You!”. Proceedings of the PharmaSUG, San Francisco, CA.
https://pharmasug.org/proceedings/2023/QT/PharmaSUG-2023-QT-233.pdf

ACKNOWLEDGMENTS

The authors would like to thank Merck management team for their review of this paper and great support to Rising Professional Club.

CONTACT INFORMATION

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

Name: Krutika Parvatikar
Enterprise: Merck & Co., Inc.
Address: 126 E. Lincoln Avenue
City, State ZIP: Rahway, NJ 07065-4607 USA
E-mail: krutika.parvatikar@merck.com

Jeff Xia
Name: Jeff Xia
Enterprise: Merck & Co., Inc.
Address: 126 E. Lincoln Avenue
City, State ZIP: Rahway, NJ 07065-4607 USA
Web: www.merck.com