PharmaSUG 2024 - Paper PO- 195

A Simple Way to Make Adaptive Pages in Listings and Tables

Yi Guo, Pfizer Inc.

ABSTRACT

Generating listings and tables is an essential skill for every statistical programmer. The task of optimizing the display of listings and tables can be a challenging one for new programmers. This is because the true length of a variable can vary significantly between patient records, consequently affecting the number of pages in the listing, file size, loading time, and empty space on a page. Similarly, in tables, the number of distinct values of a categorical variable can vary as a study matures, and summarizing such categorical variables without affecting the page break can also be challenging - ideally, we would want to summarize a categorical variable on the same page when space allows. In this paper, we provide a simple algorithm that calculates the maximum number of observations to display on each page with an aim to optimize the display, the number of pages, the file size, and the loading time.

INTRODUCTION

Pagination takes place before the PROC REPORT statement in SAS® programs. It is common practice to paginate the final work data set used in PROC REPORT by dividing the total number of observations (which we shall equate to _N_ for simplicity in this paper) by the maximum number of rows displayed per page (n), i.e., using ceil(_N_/n) in SAS®.

However, this method does not work all the time. The lengths of data values may vary widely, resulting in a different number of rows displayed across pages. In these situations, even if we adjust the value of n to avoid unwanted page breaks, we still cannot optimize the display – that is, to display as many observations as possible on each page.

Therefore, the number of observations or data records displayed per page must be flexible depending on the actual length of each data value and the maximum number of records that can fit on an individual page. In this paper, we provide a simple algorithm and demonstrate its implementation using SAS®. Not only can it automatically calculate number of observations per page, but it can also repaginate a listing or table to ensure proper page layout and numbering whenever the data or mock-up (i.e., shell or specification) is updated. It is also easily integrated into existing code for page numbering. In addition, this pagination method can allow variables with multiple categories or statistics to be fully displayed on a single page of the table, if space permits.

In the following sections, we will illustrate common scenarios where creating adaptive pages is essential, elaborate on the algorithm, and demonstrate its implementation through both listing and table examples.

COMMON SCENARIOS FOR CREATING ADAPTIVE PAGES

Below are some examples for each scenario where making adaptive pages for listings or tables is beneficial. To better exemplify these situations, we used a publicly available TFL shell template.

Scenario 1

A few records are exceptionally long, while others are short. For example, a patient has a notably lengthy entry for medical indication in Concomitant Procedures (CP) listing. This indication will wrap to multiple lines. This situation is also likely to occur in Concomitant and Prior Medications (CM) and Medical History (MH) listings.

Listing 16.2.4.7
Concomitant Procedures
All Subjects

Subject	Procedure	Indication	Start Date	Stop Date
###-###	XXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX	DDMMMYYYY	DDMMMYYYY
	XXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXX	DDMMMYYYY	ONGOING
###-###	XXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXX	DDMMMYYYY	DDMMMYYYY

Figure 1a. Long data entry for indication in CP listing.

Scenario 2

Multiple variables are concatenated into a single column in listings, causing some of the newly combined data values to have considerable length. For instance, Adverse Event (AE), System Organ Class (SOC), and Preferred Term (PT) are often combined into a single column displayed in an AE listing.

Listing 16.2.7.1
Adverse Events
All Subjects

Subject	TEAE? / Onset Dose/ SAE?	Adverse Event/ Preferred Term/ System Organ Class	Start Date/Day End Date/Day	Duration (Days)	Severity/ Relationship to Study Drug	Action Taken with Study Drug	Outcome/ Other Action Taken
###-###	YES/ 250 mg/ NO	XXXXXXXXXXXXXXXXXXXX/ XXXXXXXXXXXXXXXXXXXX/ XXXXXXXXXXXXXXXXXXXX	DDMMMYYYY/## DDMMMYYYY/##	##	XXXXXX/ XXXXXXXXXXXX	DOSE NOT CHANGED	XXXXXXX/ XXXXXXXXXXXXXX
	YES/ 1000 mg/ XXX	XXXXXXXXXXXXXXXXXXXX/ XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX/ XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX	DDMMMYYYY/## DDMMMYYYY/##	##	XXXXXX/ XXXXXXXXXXXX	DRUG INTERRUPTED Last Dose: DDMMMYYYY Restart: DDMMMYYYY	XXXXXXX/ XXXXXXXXXXXXXX

Figure 1b. Multiple variables are concatenated into one column in AE listing.

In both scenario 1 and 2, the consequence of reducing the maximum number of observations displayed per page, i.e., value of n, will be an increase in the number of pages, with a majority of them containing excess blank space.

Scenario 3

For better readability, it is preferred to have all values of a categorical variable displayed or summarized on a same page when space allows.

In Figure 1c, both variable *Race* ① and *Height at Screening* ② are very likely to stay on page 1 when there are only a few patients enrolled in this study. As more patients enrolled, if we do not manually set the page number for both variables in SAS® code, i.e., page 1 for *Race* ① and page 2 for *Height at Screening* ②, only partial statistics from variable *Height at Screening* ② will stay on page 1, with an unwanted page break in the middle (Figure 1d).

Table 14.1.2.1
Summary of Demographic and Baseline Characteristics
All Dosed Subjects

Characteristic Category/Statistic		Total (N=#)
Age at Informed Consent		
n		##
Mean		##.#
Standard Deviation		##.##
Median		##.#
Minimum		##
Maximum		##
Sex, n (%)		
Female		## (###.#)
Male		## (###.#)
Ethnicity, n (%)		
Hispanic or Latino		## (###.#)
Not Hispanic or Latino		## (###.#)
Race, n (%)		
White	①	## (###.#)
Black or African American	①	## (###.#)
Asian	①	## (###.#)
Multiple	①	## (###.#)
Height at Screening (cm)		
n	②	##
Mean	②	##.##
Standard Deviation	②	##.###
Median	②	##.##
Minimum	②	##.#
Maximum	②	##.#

Baseline is defined as the last measurement prior to the first dose of study drug.
%=100*n/N where n is the number of subjects in the specified category and N is the number of dosed subjects.

Figure 1c. The last variable is fully displayed on DM table page 1 when there are only a few patients enrolled.

Table 14.1.2.1
Summary of Demographic and Baseline Characteristics
All Dosed Subjects

Characteristic Category/Statistic		Total (N=#)
Age at Informed Consent		
n		##
Mean		##.#
Standard Deviation		##.##
Median		##.#
Minimum		##
Maximum		##
Sex, n (%)		
Female		## (###.#)
Male		## (###.#)
Ethnicity, n (%)		
Hispanic or Latino		## (###.#)
Not Hispanic or Latino		## (###.#)
Race, n (%)		
White	①	## (###.#)
Black or African American	①	## (###.#)
Asian	①	## (###.#)
American Indian or Alaskan Native	①	## (###.#)
Native Hawaiian or Other Pacific Islander	①	## (###.#)
Multiple	①	## (###.#)
Height at Screening (cm)		
n	②	##
Mean	②	##.##

Baseline is defined as the last measurement prior to the first dose of study drug.
%=100*n/N where n is the number of subjects in the specified category and N is the number of dosed subjects.

Figure 1d. Only partial statistics from the last variable are displayed on DM table page 1.

DETAILED ALGORITHM

The algorithm will be implemented in three main steps.

Step 1: After all data is processed per specification, we have the final input data set that is ready to be paginated. First, we count the number of rows for each observation that will be displayed in the output file, which in this paper we assume will be in RTF format. It is important to note that there is probably more than one variable in the data set that is considered in counting the rows. For example, in CM listing (Figure 2a), the total number of rows for the first observation is 4, and the total number of rows for the second observation is 3. Both variable *Medication Name/Preferred Term/Chemical Substance* and *Indication* should be considered in counting the rows.

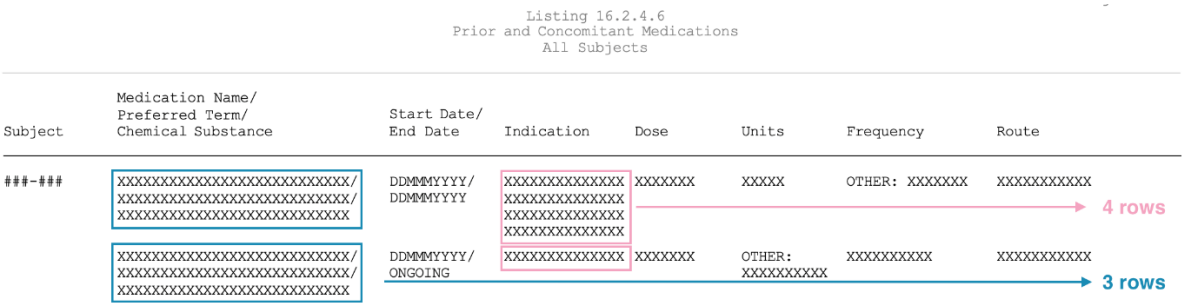

Figure 2a. Example listing that considers more than one variable in counting the rows.

Step 2: If there is an extra blank row displayed between two observations in a listing layout or between two variables in a table layout, we should add an additional row corresponding to this in the counter. Besides, each table variable may have a headline for which an extra row should also be added for page break calculation. For example:

1) + 1 row per observation when it skips one row per record.

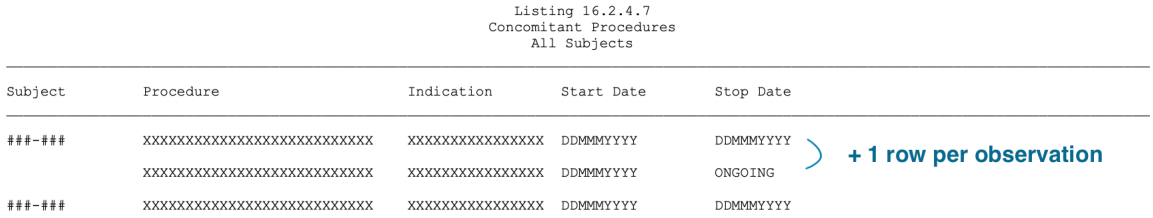

Figure 2b. Example listing that skips one row per observation.

In this example, the total number of rows for the first observation in listing is 1+1=2. Same rule applies to the following observations.

2) + 1 row per variable/group when it adds one blank row after each variable/group.
+ 1 row per variable/group since each variable/group has a headline.

Table 14.1.2.1
Summary of Demographic and Baseline Characteristics
All Dosed Subjects

Characteristic Category/Statistic	Total (N=#)
Height at Screening (cm)	
n	##
Mean	##.##
Standard Deviation	##.###
Median	##.##
Minimum	##.#
Maximum	##.#
Baseline Weight (kg)	
n	##
Mean	##.##
Standard Deviation	##.###
Median	##.##
Minimum	##.#
Maximum	##.#

+1 row per headline

+1 row per variable

Figure 2c. Example table that has a headline and adds one blank row per observation group.

In this example, the total number of rows for the first variable, i.e., Height at Screening (cm), in table is 1+6+1=8. Same rule applies to the rest of the variables.

Step 3: Finally, we perform the cumulative sum starting from the first observation until it exceeds the maximum number of rows allowed per page, i.e., value of n. Once that point is reached, we move that observation to the next page, starting at the first row, and repeat the cumulative sum process. This cycle repeats continuously until the last observation is reached.

MAKING ADAPTIVE PAGES IN LISTINGS USING SAS®

The input data set *mock_data* we use for this listing example has 8 observations and 4 variables: *USUBJID*, *COHORT*, *TXT1* and *TXT2* (Figure 3, part ①). It has already been sorted by *COHORT* and *USUBJID*. The text values have already been processed to identify wrap points demarcated by the special delimiters "`/" [a]. We assume the maximum number of rows displayed per page is 17. Only *USUBJID*, *TXT1*, and *TXT2* will be displayed in the listing.

Obs	USUBJID	COHORT	TXT1	TXT2	row_line1	row_line2	max_row_line	row_line_skip	page_lines	pageno
1	1001-001	1	xxxxxxx`/xxxxxxx`/xxxxxxx	xxxxxxx	3	1	3	4	4	1
2	1001-001	1	xxxxxxx`/xxxxxxx`/xxxxxxx`/xxxxxxx`/xxxxxxx`/xxxxxxx`/xxxxxxx`/xxxxxxx`/xxxxxxx`/xxxxxxx	xxxxxxx`/xxxxxxx`/xxxxxxx	9	3	9	10	14	1
3	1001-001	1	xxxxxxx`/xxxxxxx`/xxxxxxx`/xxxxxxx	xxxxxxx`/xxxxxxx	4	2	4	5	5	2
4	1001-002	2	xxxxxxx	xxxxxxx`/xxxxxxx`/xxxxxxx	1	3	3	4	9	2
5	1001-002	2	xxxxxxx`/xxxxxxx	xxxxxxx`/xxxxxxx	2	2	2	3	12	2
6	1001-003	2	xxxxxxx	xxxxxxx`/xxxxxxx`/xxxxxxx	1	3	3	4	16	2
7	1001-004	2	xxxxxxx`/xxxxxxx`/xxxxxxx`/xxxxxxx`/xxxxxxx	xxxxxxx	5	1	5	6	6	3
8	1001-005	3	xxxxxxx`/xxxxxxx`/xxxxxxx	xxxxxxx	3	1	3	4	10	3

① ② ③

Figure 3. Data set used for the listing example.

[a]. The ways of text wrapping can be varied, and we will not discuss further in this paper. In REFERENCES section, we provide a method that creates output with automated page breaks, proposed by Gratt, Jeremy and Tella, Aditya (2021) for your reference.

According to the DETAILED ALGORITHM section, in the first step, we count the number of rows each observation will occupy in the RTF output.

```
%let total_txt= 2; /* Total # of TXT variables in mock_data*/

data mock_data1;
       set mock_data;
       array text [*] TXT1-TXT&total_txt.;
       array rowc [*] row_line1-row_line&total_txt.;
       do i = 1 to &total_txt.;
              rowc[i] = 1+countc(text[i], '`'); /* # of rows per data value*/
       end;
       max_row_line = max(of row_line1-row_line&total_txt.);
       row_line_skip = max_row_line + 1; /* Skip one row per record */
       drop i;
run;
```

SAS® program 1: Calculating number of rows per observation for a listing.

After executing the above program, we have 4 new variables generated: *row_line1*, *row_line2*, *max_row_line* and *row_line_skip* (Figure 3, part ②).

Variable *row_line1* and *row_line2* summarize how many rows *TXT1* and *TXT2* will span per observation in the ultimate RTF file, respectively. *max_row_line* is the maximum value of *row_line1* and *row_line2*.

It is important to note that we add an additional one row to *max_row_line* since in the output file we add one blank row per observation according to the listing layout, i.e., *row_line_skip* = *max_row_line* +1.

```
data mock_data2;
       set mock_data1;
       by COHORT USUBJID;
       retain page_lines;

       if _n_ = 1 then page_lines = 0;
       page_lines + row_line_skip; /* Cumulative sum of rows */

       if _n_ = 1 then pageno = 1; /* Start at Page 1 by default */
       if page_lines > 17 then do; /* Break per 17 rows */
              page_lines = row_line_skip;
              pageno + 1;
       end;
run;
```

SAS® program 2: Numbering the page for each observation from listing.

After running the above program, we obtain 2 new variables: *page_lines* and *pageno*. And *pageno* is the outcome variable for pagination.

Variable *page_lines* is the cumulative sum of *row_line_skip* (Figure 3, part ③). Since the listing can only display no more than 17 rows per page, the first and the second observations will be shown on page 1 (i.e., 4+10=14 <17), the third observation will be displayed on page 2 (i.e., 4+10+5=19 > 17). One thing we should pay attention to is that the value of *page_lines* from the third observation is equal to the value of *row_line_skip* because it is the first record. Repeating this cycle until assigning page number to the last observation (Obs=8).

MAKING ADAPTIVE PAGES IN TABLES USING SAS®

The input data set *mock_data* has 34 observations and 4 variables: *order*, *TXT1*, *TXT2* and *VAR1* (Figure 4, part ①). We assume the maximum number of rows displayed per page is 21. Only *TXT1*, *TXT2* and *VAR1* will be displayed in the table.

The way of counting the rows in tables is slightly different from listings. Instead of counting the number of rows for each observation, we need to know the total number of rows that will be displayed in the RTF file for each variable or observation group for further calculation.

```
proc sql;
    create table mock_data1 as
    select order, TXT1, count(*) as max_row_line
    from mock_data
    group by order, TXT1;
quit;

data mock_data2;
       set mock_data1;
       by order;
       retain page_lines 0 pageno 1; /* Start at Page 1 by default */

       row_line_skip = max_row_line + 1; /* Skip one row per variable */
       row_line_skip = row_line_skip + 1; /* Each variable has a headline */

       page_lines + row_line_skip; /* Cumulative sum of rows */

       if page_lines > 21 then do; /* Break per 21 rows*/
              page_lines = row_line_skip;
              pageno + 1;
       end;
run;
```

SAS® program 3: Calculating number of rows and numbering the page for each observation group (or variable) for a table.

After executing the above program, we have the outcome data set *mock_data2* that contains 4 new variables: *max_row_line*, *row_line_skip*, *page_lines* and *pageno* (Figure 4, part ②). Here, *max_row_line* summarizes the total number of categories or statistics from *TXT2* for each unique value, e.g., Age, Sex, etc., from *TXT1*.

Particularly, in this table example, *row_line_skip* = *max_row_line*+2. See highlights from SAS® program 3. Unlike the listing example, we add 2 additional rows instead of 1 for each group. This is because each group is preceded by a headline in the table layout.

Obs	order	TXT1	TXT2	VAR1	max_row_line	row_line_skip	page_lines	pageno
1	1	Age (years)	n	##	6	8	8	1
2	1	Age (years)	Mean	##.#	6	8	8	1
3	1	Age (years)	Standard Deviation	##.##	6	8	8	1
4	1	Age (years)	Median	##.#	6	8	8	1
5	1	Age (years)	Minimum	##	6	8	8	1
6	1	Age (years)	Maximum	##	6	8	8	1
7	2	Sex, n (%)	Female	## (###.#)	2	4	12	1
8	2	Sex, n (%)	Male	## (###.#)	2	4	12	1
9	3	Ethnicity, n (%)	Hispanic or Latino	## (###.#)	2	4	16	1
10	3	Ethnicity, n (%)	Not Hispanic or Latino	## (###.#)	2	4	16	1
11	4	Race, n (%)	White	## (###.#)	6	8	8	2
12	4	Race, n (%)	Black or African American	## (###.#)	6	8	8	2
13	4	Race, n (%)	Asian	## (###.#)	6	8	8	2
14	4	Race, n (%)	American Indian or Alaskan Native	## (###.#)	6	8	8	2
15	4	Race, n (%)	Native Hawaiian or other Pacific Islander	## (###.#)	6	8	8	2
16	4	Race, n (%)	Multiple	## (###.#)	6	8	8	2
17	5	Height at Screening (cm)	n	##	6	8	16	2
18	5	Height at Screening (cm)	Mean	##.#	6	8	16	2
19	5	Height at Screening (cm)	Standard Deviation	##.##	6	8	16	2
20	5	Height at Screening (cm)	Median	##.#	6	8	16	2
21	5	Height at Screening (cm)	Minimum	##	6	8	16	2
22	5	Height at Screening (cm)	Maximum	##	6	8	16	2
23	6	Baseline Weight (kg)	n	##	6	8	8	3
24	6	Baseline Weight (kg)	Mean	##.#	6	8	8	3
25	6	Baseline Weight (kg)	Standard Deviation	##.##	6	8	8	3
26	6	Baseline Weight (kg)	Median	##.#	6	8	8	3
27	6	Baseline Weight (kg)	Minimum	##	6	8	8	3
28	6	Baseline Weight (kg)	Maximum	##	6	8	8	3
29	7	Body Mass Index (kg/m^2)	n	##	6	8	16	3
30	7	Body Mass Index (kg/m^2)	Mean	##.#	6	8	16	3
31	7	Body Mass Index (kg/m^2)	Standard Deviation	##.##	6	8	16	3
32	7	Body Mass Index (kg/m^2)	Median	##.#	6	8	16	3
33	7	Body Mass Index (kg/m^2)	Minimum	##	6	8	16	3
34	7	Body Mass Index (kg/m^2)	Maximum	##	6	8	16	3

Obs	order	TXT1	max_row_line	row_line_skip	page_lines	pageno
1	1	Age (years)	6	8	8	1
2	2	Sex, n (%)	2	4	12	1
3	3	Ethnicity, n (%)	2	4	16	1
4	4	Race, n (%)	6	8	8	2
5	5	Height at Screening (cm)	6	8	16	2
6	6	Baseline Weight (kg)	6	8	8	3
7	7	Body Mass Index (kg/m^2)	6	8	16	3

Figure 4. Data sets used for the table example.

Finally, we merge the data set ② with data set ① by variables *order* and get the analysis outcome data set that is paginated and ready for PROC REPORT (Figure 4, ① and ③).

```
data final;
       merge mock_data mock_data2;
       by order;
run;
```

SAS® program 4: Merging the data sets as the final step.

During the PROC REPORT procedure, make sure we place the new variable *pageno* at the beginning of all variables in the COLUMN statement, then define the *pageno* using `define pageno / noprint order;`, and create a break line and start a new page using `break after pageno/ page;`.

CONCLUSION

This simple algorithm is designed to work after text wrapping and before the REPORT procedure. It calculates the maximum number of observations displayed on each page in a simple way. The amazing RETAIN statement from the corresponding SAS® code keeps the value of page numbers from the current iteration of the data step to the next. It enables the program to run efficiently and therefore shorten the total lines of the code. Also, our SAS® code can be easily integrated into any existing code. There are two things we should pay attention to during the page number calculation: 1) based on the layout design, two line breaks may be added between any two observations or between two variables to enhance readability; and 2) each variable may be preceded by a headline in the table layout. So, make sure to add the correct number of additional rows to get the page numbers.

REFERENCES

DMC TFL Shells Template from ClinicalTrials.gov. Accessed February 10, 2024. https://classic.clinicaltrials.gov/ProvidedDocs/71/NCT03053271/SAP_001.pdf

Gratt, Jeremy and Tella, Aditya. (2021) “Re-pagination of ODS RTF Outputs to Automate Page Breaks and Minimize Splits Across Pages.” PharmaSUG 2021, Paper AP-101.

SAS Institute Inc. 2016. SAS® 9.4 DATA Step Statements: Reference. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2019. SAS® Certified Professional Prep Guide: Advanced Programming Using SAS® 9.4. Cary, NC: SAS Institute Inc.

ACKNOWLEDGEMENT

The author would like to thank Kuldeep Sen and Michiel Hagendoorn for their feedback, support, and guidance.

CONTACT INFORMATION

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

Yi Guo
Pfizer Inc.
yi.guo@pfizer.com