PharmaSUG 2025 - Paper OS-077

Comparing SAS® and R Approaches in Reshaping data

Yachen Wang, Chen Ling, AbbVie Inc.

ABSTRACT

Data reshaping is a fundamental process in data management. Both SAS® and R offer robust capabilities to transform data between wide and long formats, enabling researchers to manipulate and reorganize datasets for effective analysis. This paper compares SAS® **PROC TRANSPOSE** and R's **pivot_longer()** and **pivot_wider()** from **{tidyr}** package in data reshaping.

Apart from their basic data transformation capabilities, both tools can customize variable names, handle duplications and deal with unused variables. SAS® is more preferred while handling labels in data reshaping, due to its unique labeling feature. R, on the other hand, offers additional flexibility with its useful options, enhancing the ability to manage more complex data scenarios.

Practical example codes in SAS® and R are provided to illustrate these features and differences. These examples will help users understand the application of each method and demonstrate the strengths and limitations of each approach. By examining these approaches and providing illustrative examples, we provide insights on selecting the appropriate tool for various data transformation tasks in data analysis.

INTRODUCTION

Before doing statistical analysis, it's very important to do data wrangling and get the tidy data. Tidy data exists in two forms: wide data and long data. Wide data is a format in which each subject (or observational unit) has a single row, and different variables (measurements) are presented in separate columns. Long data, on the other hand, a format in which each row represents a single measurement or observation, often includes a variable indicating the time or condition of the measurement. The choice of whether to display the data in wide or long format will not change the information stored in the dataset, however, different statistical analysis may require different data setup.

Both SAS and R are powerful tools for reshaping data. SAS® is a well-established platform known for its robust data handling capabilities and is heavily utilized in the pharmaceutical industry (Ling & Wang, 2025). ***PROC TRANSPOSE*** is one of its key procedures for reshaping data, allowing users to pivot data from wide to long and vice versa, while maintaining the integrity of variable labels. On the other hand, R offers a more flexible and versatile approach to organize data through **{tidyr}** package (Ling & Wang, 2025). Its ***pivot_longer()*** and ***pivot_wider()*** functions are specifically designed for reshaping data and provide an extensive set of options for handling complex data structures.

This paper presents a comparative analysis of these reshaping techniques in SAS® and R. By examining common features, unique capabilities, and practical use cases, we aim to provide a clear understanding of how each tool can be effectively used in data transformation tasks. Practical example codes in both SAS® and R also illustrate the application of these methods and highlight their respective strengths and limitations. Through this comparison, we intend to guide data practitioners in selecting the appropriate tool for their specific data reshaping needs, ensuring efficient and accurate data analysis processes.

SOURCE DATA

The source data are dummy datasets including 1) STUDYID, USUBJID, TRT01AN, TRT01A and SEX in a dummy ADSL 2) STUDYID, USUBJID, TRTA, AEBODSYS and AEDECOD in a dummy ADAE.

FROM ROW TO COLUMN (WIDE-TO-LONG FORMAT)

Table 1 below is a sample ADSL that is in a wide format, having treatment of period information in one row. This data could be arranged in a long format, which has treatment of period information in separated rows.

STUDYID	USUBJID	TRT01A	TRT01AN	TRT02A	TRT02AN
STUDY-A	1010	REGIMEN B	2	REGIMEN D	4
STUDY-A	1011	REGIMEN A	1	REGIMEN B	2
STUDY-A	1019	REGIMEN A	1	REGIMEN B	2
STUDY-A	1021	REGIMEN A	1	REGIMEN B	2
STUDY-A	1022	REGIMEN B	2	REGIMEN D	4
STUDY-A	1029	REGIMEN B	2	REGIMEN D	4
STUDY-A	1040	REGIMEN B	2		

Figure 1. Sample ADSL

COMMON FEATURES IN SAS AND R

Basic Use

In SAS, **PROC TRANSPOSE** is designed to achieve this, variables in `VAR` options contain the actual data that needs to be transposed and BY variables are those not transposed. The dataset needs to be sorted by the "BY variables" before running **PROC TRANSPOSE** (not sorting will cause an error).

```
proc sort data=adsl;
  by STUDYID USUBJID;
run;
proc transpose data=adsl out=sasadsl1;
  by STUDYID USUBJID;
  var TRT01AN TRT02AN;
run;
```

SAS program 1. PROC TRANSPOSE basic use to get long data

STUDYID	USUBJID	_NAME_	_LABEL_	COL1
STUDY-A	1010	TRT01AN	Actual Treatment for Period 01 (N)	2
STUDY-A	1010	TRT02AN	Actual Treatment for Period 02 (N)	4
STUDY-A	1011	TRT01AN	Actual Treatment for Period 01 (N)	1
STUDY-A	1011	TRT02AN	Actual Treatment for Period 02 (N)	2
STUDY-A	1019	TRT01AN	Actual Treatment for Period 01 (N)	1
STUDY-A	1019	TRT02AN	Actual Treatment for Period 02 (N)	2
STUDY-A	1021	TRT01AN	Actual Treatment for Period 01 (N)	1
STUDY-A	1021	TRT02AN	Actual Treatment for Period 02 (N)	2
STUDY-A	1022	TRT01AN	Actual Treatment for Period 01 (N)	2
STUDY-A	1022	TRT02AN	Actual Treatment for Period 02 (N)	4
STUDY-A	1029	TRT01AN	Actual Treatment for Period 01 (N)	2
STUDY-A	1029	TRT02AN	Actual Treatment for Period 02 (N)	4
STUDY-A	1040	TRT01AN	Actual Treatment for Period 01 (N)	2
STUDY-A	1040	TRT02AN	Actual Treatment for Period 02 (N)	.

Figure 2. PROC TRANSPOSE basic use to get long data

In R, **pivot_longer()** function in the **{tidyr}** package is used to transform the dataset into long format. Variables in `cols` are the columns to pivot into longer format. One important detail to consider is that the variables in the "`cols`" option must be of the same type. If they are not, R will generate an error (Figure 4).

```
radsl1<-adsl%>%select(STUDYID,USUBJID,TRT01AN,TRT02AN)%>%
        pivot_longer(cols=c("TRT01AN","TRT02AN"))
```

R program 2. pivot_longer basic use

STUDYID Study Identifier	USUBJID Subject Identifier for the Study	name	value
STUDY-A	1010	TRT01AN	2
STUDY-A	1010	TRT02AN	4
STUDY-A	1011	TRT01AN	1
STUDY-A	1011	TRT02AN	2
STUDY-A	1019	TRT01AN	1
STUDY-A	1019	TRT02AN	2
STUDY-A	1021	TRT01AN	1
STUDY-A	1021	TRT02AN	2
STUDY-A	1022	TRT01AN	2
STUDY-A	1022	TRT02AN	4
STUDY-A	1029	TRT01AN	2
STUDY-A	1029	TRT02AN	4
STUDY-A	1040	TRT01AN	2
STUDY-A	1040	TRT02AN	NA

Figure 3. transfer ADSL using pivot_longer basic use

```
> radsl1<-adsl%>%select(STUDYID,USUBJID,TRT01AN,TRT01A)%>%pivot_longer(cols=c("TRT01AN","TRT01A"))
Error in `pivot_longer()`:
! Can't combine `TRT01AN` <double> and `TRT01A` <character>.
Run `rlang::last_trace()` to see where the error occurred.
```

Figure 4. error when using different types in "cols" option

Customize variable names

In **PROC TRANSPOSE,** there are optional arguments to customize variable names. `Prefix` and `suffix` specify a prefix/suffix to use in constructing names for transposed variables, and `name` specify the name for the variable being transposed. These arguments can enhance the appearance and clarity of the output dataset.

```
proc sort data=adsl;
 by STUDYID USUBJID;
run;
proc transpose data=adsl out=sasadsl2 prefix=PRE suffix=SUF name=TRTANAM;
 by STUDYID USUBJID;
 var TRT01AN TRT02AN;
run;
```

SAS program 2. customize variable names in PROC TRANSPOSE

	STUDYID	USUBJID	TRTANAM	_LABEL_	PRE1SUF
1	STUDY-A	1010	TRT01AN	Actual Treatment for Period 01 (N)	2
2	STUDY-A	1010	TRT02AN	Actual Treatment for Period 02 (N)	4
3	STUDY-A	1011	TRT01AN	Actual Treatment for Period 01 (N)	1
4	STUDY-A	1011	TRT02AN	Actual Treatment for Period 02 (N)	2
5	STUDY-A	1019	TRT01AN	Actual Treatment for Period 01 (N)	1
6	STUDY-A	1019	TRT02AN	Actual Treatment for Period 02 (N)	2
7	STUDY-A	1021	TRT01AN	Actual Treatment for Period 01 (N)	1
8	STUDY-A	1021	TRT02AN	Actual Treatment for Period 02 (N)	2
9	STUDY-A	1022	TRT01AN	Actual Treatment for Period 01 (N)	2
10	STUDY-A	1022	TRT02AN	Actual Treatment for Period 02 (N)	4
11	STUDY-A	1029	TRT01AN	Actual Treatment for Period 01 (N)	2
12	STUDY-A	1029	TRT02AN	Actual Treatment for Period 02 (N)	4
13	STUDY-A	1040	TRT01AN	Actual Treatment for Period 01 (N)	2
14	STUDY-A	1040	TRT02AN	Actual Treatment for Period 02 (N)	.

Figure 5. customize variable names in PROC TRANSPOSE

Meanwhile in R**, pivot_longer()** function also provides capability to customize the name of the transposed columns. The "`names_to`" argument is used to specify the name of new column(s) to create from the information stored in the column names of data specified by "*cols*", and "`values_to`" argument allows users to specify the name of the column to be created from the data stored in cell values.

```
radsl1name<-adsl%>%select(STUDYID,USUBJID,TRT01AN,TRT02AN)%>%
        pivot_longer(cols=c("TRT01AN","TRT02AN"),names_to="TRTNAME",values_to= "TRTNUM")
```

R program 2. customize variable names in pivot_longer

	STUDYID Study Identifier	USUBJID Subject Identifier for the Study	TRTNAME	TRTNUM
1	STUDY-A	1010	TRT01AN	2
2	STUDY-A	1010	TRT02AN	4
3	STUDY-A	1011	TRT01AN	1
4	STUDY-A	1011	TRT02AN	2
5	STUDY-A	1019	TRT01AN	1
6	STUDY-A	1019	TRT02AN	2
7	STUDY-A	1021	TRT01AN	1
8	STUDY-A	1021	TRT02AN	2
9	STUDY-A	1022	TRT01AN	2
10	STUDY-A	1022	TRT02AN	4
11	STUDY-A	1029	TRT01AN	2
12	STUDY-A	1029	TRT02AN	4
13	STUDY-A	1040	TRT01AN	2
14	STUDY-A	1040	TRT02AN	NA

Figure 6. customize variable names in pivot_longer

FEATURES IN SAS BUT NOT IN R

Label

As illustrated in Figure 2 and Figure 5, variable labels are transferred into a column named "_LABEL_", this column can be renamed by using "`label=`" option in **PROC TRANSPOSE** procedure in SAS.

```
proc sort data=adsl;
 by STUDYID USUBJID;
run;
proc transpose data=adsl out=sasadsl2 prefix=PRE suffix=SUF name=TRTANAM label=TRTALABEL;
 by STUDYID USUBJID;
 var TRT01AN TRT02AN;
run;
```

SAS program 3. customize _LABEL_ column in PROC TRANSPOSE

	STUDYID	USUBJID	TRTANAM	TRTALABEL	PRE1SUF
1	STUDY-A	1010	TRT01AN	Actual Treatment for Period 01 (N)	2
2	STUDY-A	1010	TRT02AN	Actual Treatment for Period 02 (N)	4
3	STUDY-A	1011	TRT01AN	Actual Treatment for Period 01 (N)	1
4	STUDY-A	1011	TRT02AN	Actual Treatment for Period 02 (N)	2
5	STUDY-A	1019	TRT01AN	Actual Treatment for Period 01 (N)	1
6	STUDY-A	1019	TRT02AN	Actual Treatment for Period 02 (N)	2
7	STUDY-A	1021	TRT01AN	Actual Treatment for Period 01 (N)	1
8	STUDY-A	1021	TRT02AN	Actual Treatment for Period 02 (N)	2
9	STUDY-A	1022	TRT01AN	Actual Treatment for Period 01 (N)	2
10	STUDY-A	1022	TRT02AN	Actual Treatment for Period 02 (N)	4
11	STUDY-A	1029	TRT01AN	Actual Treatment for Period 01 (N)	2
12	STUDY-A	1029	TRT02AN	Actual Treatment for Period 02 (N)	4
13	STUDY-A	1040	TRT01AN	Actual Treatment for Period 01 (N)	2
14	STUDY-A	1040	TRT02AN	Actual Treatment for Period 02 (N)	.

Figure 7. customize label names in PROC TRANSPOSE

FEATURES IN R BUT NOT IN SAS

Grouped variable

One powerful option in **pivot_longer()** is `names_pattern`, which assists users in transforming grouped variables with similar patterns. For example, treatment name and treatment number can be transformed together as a group in provided ADSL (Figure 1) by using "`names_pattern=`" option. The letters put in this

option follow the Regular Expression in R (e.g. "//d" means digit), and `names_to`, under this circumstance, requires names to be provided for all new columns.

```
radsl3<-adsl%>%select(STUDYID,USUBJID,TRT01AN,TRT01A,TRT02AN,TRT02A)%>%
        pivot_longer(cols = starts_with("TRT"),
        names_to = c("treatment", ".value"),
        names_pattern = "TRT(\\d+)(AN|A)")
```

R program 3. names_pattern in pivot_longer

STUDYID Study Identifier	USUBJID Subject Identifier for the Study	treatment	AN	A
STUDY-A	1010	01	2	REGIMEN B
STUDY-A	1010	02	4	REGIMEN D
STUDY-A	1011	01	1	REGIMEN A
STUDY-A	1011	02	2	REGIMEN B
STUDY-A	1019	01	1	REGIMEN A
STUDY-A	1019	02	2	REGIMEN B
STUDY-A	1021	01	1	REGIMEN A
STUDY-A	1021	02	2	REGIMEN B
STUDY-A	1022	01	2	REGIMEN B
STUDY-A	1022	02	4	REGIMEN D
STUDY-A	1029	01	2	REGIMEN B
STUDY-A	1029	02	4	REGIMEN D
STUDY-A	1040	01	2	REGIMEN B
STUDY-A	1040	02	NA	

Figure 8. names_pattern in pivot_longer

Other useful functions

pivot_longer() function also have other useful option, for example `values_drop_na` will drop rows that contain only NAs in the `value_to` column, `names_transform` and `values_transform` can change the types of specific columns. R program 4 below is from R program 2 by adding `values_drop_na` and `values_transform`. Figure 9 illustrates that missing treatment is removed and TRTNUM variable is converted to a character type.

```
radsl1name<-adsl%>%select(STUDYID,USUBJID,TRT01AN,TRT02AN)%>%
         pivot_longer(cols=c("TRT01AN","TRT02AN"),names_to="TRTNAME",values_to= "TRTNUM",
         values_drop_na = TRUE, values_transform = list(TRTNUM = as.character))
```

R program 4. useful function in pivot_longer

STUDYID Study Identifier	USUBJID Subject Identifier for the Study	TRTNAME	TRTNUM
STUDY-A	1010	TRT01AN	2
STUDY-A	1010	TRT02AN	4
STUDY-A	1011	TRT01AN	1
STUDY-A	1011	TRT02AN	2
STUDY-A	1019	TRT01AN	1
STUDY-A	1019	TRT02AN	2
STUDY-A	1021	TRT01AN	1
STUDY-A	1021	TRT02AN	2
STUDY-A	1022	TRT01AN	2
STUDY-A	1022	TRT02AN	4
STUDY-A	1029	TRT01AN	2
STUDY-A	1029	TRT02AN	4
STUDY-A	1040	TRT01AN	2

Figure 9. useful function in pivot_longer

FROM COLUMN TO ROW (LONG-TO-WIDE FORMAT)

COMMON FEATURES IN SAS AND R

Basic Use

In SAS if we want to rearrange data from long format to wide format, we also need **PROC TRANSPOSE** procedure. Data from Figure 2 is used in this section. First, the ADSL data is sorted by STUDYID, USUBJID, then **PROC TRANSPOSE** is employed, where the `id` statement specifies the new variable name, and the "`var`" statement indicates the source of the values to fill those variables.

```
proc sort data=sasadsl1;
 by STUDYID USUBJID;
run;
proc transpose data=sasadsl1 out=sasadsl3;
 by STUDYID USUBJID ;
 id _NAME_;
 var COL1;
run;
```

SAS program 4. PROC TRANSPOSE basic use to get wide data

STUDYID	USUBJID	_NAME_	TRT01AN	TRT02AN
STUDY-A	1010	COL1	2	4
STUDY-A	1011	COL1	1	2
STUDY-A	1019	COL1	1	2
STUDY-A	1021	COL1	1	2
STUDY-A	1022	COL1	2	4
STUDY-A	1029	COL1	2	4
STUDY-A	1040	COL1	2	

Figure 10. PROC TRANSPOSE basic use to get wide data

In R, the **pivot_wider()** function can be used to get wide format dataset. For instance, using the data from Figure 3, columns names will be specified using the `names_from` argument, and the columns values will be specified using the `values_from` argument.

```
radsl3<-radsl1%>%pivot_wider(names_from = name, values_from = value)
```

R program 5. pivot_wider basic use

STUDYID Study Identifier	USUBJID Subject Identifier for the Study	TRT01AN	TRT02AN
STUDY-A	1010	2	4
STUDY-A	1011	1	2
STUDY-A	1019	1	2
STUDY-A	1021	1	2
STUDY-A	1022	2	4
STUDY-A	1029	2	4
STUDY-A	1040	2	NA

Figure 11. pivot_wider basic use

Customize variable names

While rearranging data from long format to wide format, the `prefix` and `suffix` options in SAS **PROC TRANSPOSE** procedure are also applicable, as discussed in the previous section.

Similarly in R, the `names_prefix` option in **pivot_wider** function allows you to add a string to the beginning of each variable name in the wide dataset. For example, R Program 6 demonstrates how to modify R Program 5 by adding a "TT" prefix to the newly created column names.

```
radsl3<-radsl1%>%pivot_wider(names_from = name, values_from = value, names_prefix = "TT")
```

R program 6. prefix in pivot_wider

STUDYID Study Identifier	USUBJID Subject Identifier for the Study	TTTRT01AN	TTTRT02AN
STUDY-A	1010	2	4
STUDY-A	1011	1	2
STUDY-A	1019	1	2
STUDY-A	1021	1	2
STUDY-A	1022	2	4
STUDY-A	1029	2	4
STUDY-A	1040	2	NA

Figure 12. prefix in pivot_wider

Handle Duplication

Sometimes, unfortunately, ID value may not be unique within a BY group. For example, in Figure 13 the _NAME_ is not unique within the STUDYID and USUBJID. To reshape the data in such cases, the `let` option in the **PROC TRANSPOSE** procedure allows for duplicate ID values. However, it is important to note that this option will only transpose the last occurrence of the particular `ID` value within the dataset or `BY` group (Harrington, 2024). (Figure 14).

	STUDYID	USUBJID	_NAME_	_LABEL_	COL1
1	STUDY-A	1010	TRTAN	Actual Treatment for Period 01 (N)	2
2	STUDY-A	1010	TRTAN	Actual Treatment for Period 02 (N)	4
3	STUDY-A	1011	TRTAN	Actual Treatment for Period 01 (N)	1
4	STUDY-A	1011	TRTAN	Actual Treatment for Period 02 (N)	2
5	STUDY-A	1019	TRTAN	Actual Treatment for Period 01 (N)	1
6	STUDY-A	1019	TRTAN	Actual Treatment for Period 02 (N)	2
7	STUDY-A	1021	TRTAN	Actual Treatment for Period 01 (N)	1
8	STUDY-A	1021	TRTAN	Actual Treatment for Period 02 (N)	2
9	STUDY-A	1022	TRTAN	Actual Treatment for Period 01 (N)	2
10	STUDY-A	1022	TRTAN	Actual Treatment for Period 02 (N)	4
11	STUDY-A	1029	TRTAN	Actual Treatment for Period 01 (N)	2
12	STUDY-A	1029	TRTAN	Actual Treatment for Period 02 (N)	4
13	STUDY-A	1040	TRTAN	Actual Treatment for Period 01 (N)	2
14	STUDY-A	1040	TRTAN	Actual Treatment for Period 02 (N)	.

Figure 13

```
proc sort data=sasadsl1;
 by STUDYID USUBJID;
run;
proc transpose data=sasadsl1 out=sasadsl4 let;
 by STUDYID USUBJID ;
 id _NAME_;
 var COL1;
run;
```

SAS program 5. using “let” in Proc transpose

STUDYID	USUBJID	_NAME_	TRTAN
STUDY-A	1010	COL1	4
STUDY-A	1011	COL1	2
STUDY-A	1019	COL1	2
STUDY-A	1021	COL1	2
STUDY-A	1022	COL1	4
STUDY-A	1029	COL1	4
STUDY-A	1040	COL1	

Figure 14. using “let” in PROC TRANSPOSE

If “let” is not in this procedure, SAS will give an error.

```
ERROR: The ID value "TRTAN" occurs twice in the same BY group.
NOTE: The above message was for the following BY group:
      Study Identifier=STUDY-A Subject Identifier for the Study=1010
```

Figure 15. error for not unique variable

Unlike SAS, **pivot_wider()** in R is equipped with the ability to transfer all records into wide format, even though the combination of `id_cols` and `names_from` columns does not uniquely identify an observation. This is achieved by using the `values_fn = list()` argument. While summarizing adverse events with its associated subject number by treatment in a sample dummy ADAE (Figure 16), `values_fn = list` can be utilized in **pivot_wider()** function (as shown in R program 7). This time the duplicated subjects are not deleted, instead R stored the subject numbers in a list (Figure 17).

STUDYID Study Identifier	USUBJID Subject Identifier for the Study	TRTAN Actual Treatment (N)	AEBODSYS Body System or Organ Class	AEDECOD Dictionary-Derived Term
STUDY-A	1017	2	Infections and infestations	Upper respiratory tract infection
STUDY-A	1017	1	Gastrointestinal disorders	Eructation
STUDY-A	1029	1	Nervous system disorders	Headache
STUDY-A	1040	2	Nervous system disorders	Dysgeusia
STUDY-A	1040	2	Skin and subcutaneous tissue disorders	Ingrown hair
STUDY-A	1042	2	Infections and infestations	Rhinitis
STUDY-A	1069	2	Nervous system disorders	Headache
STUDY-A	1069	2	Reproductive system and breast disorders	Dysmenorrhoea
STUDY-A	1070	1	Musculoskeletal and connective tissue disorders	Back pain
STUDY-A	1070	2	Skin and subcutaneous tissue disorders	Dermatitis contact
STUDY-A	1071	2	Infections and infestations	Conjunctivitis
STUDY-A	1071	1	Skin and subcutaneous tissue disorders	Dermatitis contact
STUDY-A	1071	1	Nervous system disorders	Headache

Figure 16

```
radae1<-adae<-pivot_wider(names_from=TRTAN,values_from=USUBJID,names_prefix =
"REG",values_fn = list)
```

R program 7. values_fn in pivot_wider

AEBODSYS Body System or Organ Class	AEDECOD Dictionary-Derived Term	REGIMEN B	REGIMEN A
Infections and infestations	Upper respiratory tract infection	1017	NULL
Gastrointestinal disorders	Eructation	NULL	1017
Nervous system disorders	Headache	1069	c("1029", "1071")
Nervous system disorders	Dysgeusia	1040	NULL
Skin and subcutaneous tissue disorders	Ingrown hair	1040	NULL
Infections and infestations	Rhinitis	1042	NULL
Reproductive system and breast disorders	Dysmenorrhoea	1069	NULL
Musculoskeletal and connective tissue disorders	Back pain	NULL	1070
Skin and subcutaneous tissue disorders	Dermatitis contact	1070	1071
Infections and infestations	Conjunctivitis	1071	NULL

Figure 17

Keep non-transposed columns

In SAS, to keep other non-transposed columns which are not in BY group, "`copy`" in the **PROC TRANSPOSE** will be helpful. For example, to retain the sex variable in the dataset (Figure 18), you can include it in the *copy* statement. Note that `copy` statement copies variables directly from the input data to output data. This process does not alter the number of observations, which may lead to missing values in the transposed variables, as illustrated in Figure 19.

STUDYID	USUBJID	_NAME_	_LABEL_	COL1	SEX
STUDY-A	1010	TRT01AN	Actual Treatme...	2	M
STUDY-A	1010	TRT02AN	Actual Treatme...	4	M
STUDY-A	1011	TRT01AN	Actual Treatme...	1	M
STUDY-A	1011	TRT02AN	Actual Treatme...	2	M
STUDY-A	1019	TRT01AN	Actual Treatme...	1	M
STUDY-A	1019	TRT02AN	Actual Treatme...	2	M
STUDY-A	1021	TRT01AN	Actual Treatme...	1	F
STUDY-A	1021	TRT02AN	Actual Treatme...	2	F
STUDY-A	1022	TRT01AN	Actual Treatme...	2	F
STUDY-A	1022	TRT02AN	Actual Treatme...	4	F
STUDY-A	1029	TRT01AN	Actual Treatme...	2	F
STUDY-A	1029	TRT02AN	Actual Treatme...	4	F
STUDY-A	1040	TRT01AN	Actual Treatme...	2	M
STUDY-A	1040	TRT02AN	Actual Treatme...	.	M

Figure 18

```
proc transpose data=adsl1s out=adsl3s;
 by STUDYID USUBJID;
 id _NAME_;
 var COL1;
 copy SEX;
run;
```

SAS program 6. using “copy” in PROC TRANSPOSE

STUDYID	USUBJID	SEX	_NAME_	TRT01AN	TRT02AN
STUDY-A	1010	M	COL1	2	4
STUDY-A	1010	M		.	.
STUDY-A	1011	M	COL1	1	2
STUDY-A	1011	M		.	.
STUDY-A	1019	M	COL1	1	2
STUDY-A	1019	M		.	.
STUDY-A	1021	F	COL1	1	2
STUDY-A	1021	F		.	.
STUDY-A	1022	F	COL1	2	4
STUDY-A	1022	F		.	.
STUDY-A	1029	F	COL1	2	4
STUDY-A	1029	F		.	.
STUDY-A	1040	M	COL1	2	.
STUDY-A	1040	M		.	.

Figure 19

In R, unlike in SAS, non-transposed columns are automatically retained during the transformation process. By using R Program 4 on data Figure 20, the resulting output will be as depicted in Figure 21.

STUDYID Study Identifier	**USUBJID** Subject Identifier for the Study	**name**	**value**	**sex**
STUDY-A	1010	TRT01AN	2	M
STUDY-A	1010	TRT02AN	4	M
STUDY-A	1011	TRT01AN	1	M
STUDY-A	1011	TRT02AN	2	M
STUDY-A	1019	TRT01AN	1	M
STUDY-A	1019	TRT02AN	2	M
STUDY-A	1021	TRT01AN	1	F
STUDY-A	1021	TRT02AN	2	F
STUDY-A	1022	TRT01AN	2	F
STUDY-A	1022	TRT02AN	4	F
STUDY-A	1029	TRT01AN	2	F
STUDY-A	1029	TRT02AN	4	F
STUDY-A	1040	TRT01AN	2	M
STUDY-A	1040	TRT02AN	*NA*	M

Figure 20

STUDYID Study Identifier	USUBJID Subject Identifier for the Study	sex	TRT01AN	TRT02AN
STUDY-A	1010	M	2	4
STUDY-A	1011	M	1	2
STUDY-A	1019	M	1	2
STUDY-A	1021	F	1	2
STUDY-A	1022	F	2	4
STUDY-A	1029	F	2	4
STUDY-A	1040	M	2	NA

Figure 21

FEATURES IN SAS BUT NOT IN R

Label

Handling labels is an important feature of SAS (Wang & Ling, 2025), we can even add labels to the transposed variables. If label is already in the origin dataset as a variable, "`idlabel`" can be used to add label for the new variables. In the following example, data from Figure 2 is used for demonstration.

```
proc sort data=sasadsl1;
 by STUDYID USUBJID;
run;
proc transpose data=sasadsl1 out=sasadsl4;
 by STUDYID USUBJID ;
 id _NAME_;
 var COL1;
 idlabel _LABEL_;
run;
```

SAS program 7. using "idlabel" in PROC TRANSPOSE

Name	Type	Length	Format	Informat	Label
STUDYID	Character	7			Study Identifier
USUBJID	Character	4			Subject Identifier for the Study
NAME	Character	8			NAME OF FORMER VARIABLE
TRT01AN	Numeric	8			Actual Treatment for Period 01 (N)
TRT02AN	Numeric	8			Actual Treatment for Period 02 (N)

Figure 22 attributes of variables in the transposed dataset

FEATURES IN R BUT NOT IN SAS

Names expand

There is a useful option in **pivot_longer()** named `names_expand`. With this option, the output will contain column names corresponding to a complete expansion of all possible values in `names_from`. If the study in Figure 3 has three treatment periods and the dataset currently lacks information for treatment 3, but we want to display treatment 3 as missing in final output dataset. We can define the treatment column as a factor in R, then using `names_expand = TRUE` to get a TRT03AN in the final dataset (Figure 23).

```
radsl1$name <- factor(c("TRT01AN", "TRT02AN", "TRT03AN"))
radsl3expand<-radsl1%>%
              pivot_wider(names_from = name, values_from = value,names_expand = TRUE)
```

R program 8. names_expand in pivot_wider

	STUDYID Study Identifier	USUBJID Subject Identifier for the Study	TRT01AN	TRT02AN	TRT03AN
1	STUDY-A	1010	2	4	NA
2	STUDY-A	1011	1	2	NA
3	STUDY-A	1019	1	2	NA
4	STUDY-A	1021	1	2	NA
5	STUDY-A	1022	2	4	NA
6	STUDY-A	1029	2	4	NA
7	STUDY-A	1040	2	NA	NA
8	STUDY-A	1041	1	2	NA
9	STUDY-A	1042	2	4	NA

Figure 23

Fill missing value

The `values_fill` option in **pivot_wider()** function specifies what value should be filled in when missing. It's important to note that this option only works when the record doesn't exist. If the NA is already in the dataset, it will remain unchanged. Figure 24 displays the outcome of applying R program 9 to the data in Figure 6 and Figure 9, respectively.

```
rasl2fill1<-radsl1%>%
       pivot_wider(names_from=TRTNAME,values_from=TRTNUM,values_fill= list(TRTNUM = 0))
rasl2fill<-radsl2%>%
       pivot_wider(names_from=TRTNAME,values_from=TRTNUM,values_fill= list(TRTNUM = "0"))
```

R program 9. names_expand in pivot_wider

STUDYID Study Identifier	USUBJID Subject Identifier for the Study	TRT01AN	TRT02AN
STUDY-A	1010	2	4
STUDY-A	1011	1	2
STUDY-A	1019	1	2
STUDY-A	1021	1	2
STUDY-A	1022	2	4
STUDY-A	1029	2	4
STUDY-A	1040	2	NA

STUDYID Study Identifier	USUBJID Subject Identifier for the Study	TRT01AN	TRT02AN
STUDY-A	1010	2	4
STUDY-A	1011	1	2
STUDY-A	1019	1	2
STUDY-A	1021	1	2
STUDY-A	1022	2	4
STUDY-A	1029	2	4
STUDY-A	1040	2	0

Figure 24

OVERALL COMPARISON

The following table makes a comparison between **PROC TRANSPOSE** and **{tidyr} package** in R from different perspectives.

#	Aspect	PROC TRANSPOSE	{tidyr} package
1	Language Style	Procedure-based	Function-based
2	Requirement	Data sorted	Package installed
3	Code Structure	proc transpose data=dataset; by variable; run;	data%>%pivot_longer(cols=columns[to longer format]) data%>%pivot_wider(names_from=name)
4	Scope	Handle both wide and long dataset	pivot_wider() for long-to-wide format pivot_longer() for wide-to-long format
5	Strength	Can handle labels	More options (e.g handle missing value and grouped data etc.)

6	Limitation	Can't transfer duplication	Variables in "cols=" option need to be same type

Table 1

CONCLUSION

Both SAS® **PROC TRANSPOSE** and R **{tidyr}** package are powerful tools for data manipulation, each with its own advantages and limitations. The decision to use one over the other often depends on factors such as data structure, user familiarity and the specific requirements of the organization. In this paper, we present examples using both SAS® **PROC TRANSPOSE** as well as **pivot_longer()** and **pivot_wider()** in R **{tidyr}** ,comparing them from various angles. We aim to provide readers with more valuable references when choosing between SAS® and R in the data reshaping process.

REFERENCES

Harrington, T. J. (2024). An Introduction to the SAS® Transpose Procedure and its Options. Baltimore, MD. Retrieved from PharmaSUG 2024: https://www.lexjansen.com/pharmasug/2024/AP/PharmaSUG-2024-AP-138.pdf

Ling, C., & Wang, Y. (2025). TLFQC: A High-compatible R Shiny based Platform for Automated and Codeless TLFs Generation and Validation. *PharmaSUG 2025 conference proceedings.* San Diego.

Ling, C., & Wang, Y. (2025). Writing SAS MACROs in R? R functions can help! *PharmaSUG 2025 conference proceedings.* San Diego, CA.

Wang, Y., & Ling, C. (2025). Controlling attributes of .xpt files generated by R. *PharmaSUG 2025 conference proceedings.* San Diego, CA.

ACKNOWLEDGMENTS

We would like to express our sincere gratitude to Rina Loke, Christelle Raynaud and the R team, Xiangdong Zhou, Holly Peterson and Ujjwala Powers for their support, trust, and encouragement all along.

CONTACT INFORMATION

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

Yachen Wang
AbbVie Inc.
Yachen.wang@abbvie.com

Chen Ling
AbbVie Inc.
Chen.ling@abbvie.com