

Introduction

Data integrity is critical in the Pharmaceutical Industry to guarantee that products meet required quality standards, and satisfy regulatory rules. Therefore, the validation of clinical trial datasets, analyses and output is essential to safeguard for accuracy and consistency.

Much of the industry uses SAS, with independent dual programming and SAS Proc Compare to help them to validate datasets, analyses and output. However, as more people are exploring and experimenting generating their datasets, analyses and output in R, how would you do a ‘Proc Compare’ in R.

This poster compares three available data frame comparison functions in R against a SAS Proc Compare. It points out how a programmer could utilise them to help validate a data frame and it seeks to identify appropriate R functions that provide the flexibility and usability of a Proc Compare in SAS.

Conclusion

This poster highlighted three compare functions in R. Each proved to effectively validate data frames, offering a lot of the same functionality as a SAS Proc Compare. Consequently, it can be concluded that R is equipped with at least three good alternatives to a Proc Compare, and possibly more as this is not an exhaustive list.

So, with more than one option available, which function do you prefer to use and why?

SAS Proc Compare

What is it? Proc Compare is a SAS procedure that allows two datasets to be compared for properties, number of observations, number of variables, variable content, formatting and much more. The below syntax and output illustrate a basic comparison using Proc Compare to assist in validating two datasets, a task that Statistical Programmers are routinely expected to complete.

- Selected Key Features:**
- Sub-setting statement **WHERE**
 - Control the output with **MAXPRINT=**
 - LISTALL** option to view any variables and observations not common to both datasets
 - Compare metadata of two datasets **NOVALUES, LISTVAR**
 - Fine tune difference tolerance with **METHOD= / CRITERION=** options
 - Limit comparison to predetermined variables only
 - Create in-depth record level comparison with a common ID variable, and **TRANPOSE**
 - Use **OUT** arguments **OUTCOMP, OUTDIF, OUTNOEQUAL** to create output dataset of your comparison
 - Compare two variables within a single dataset using the **VAR** and **WITH** statements
 - Shorten the output with **NOVALUES, BRIEF** options
 - Handle missing values **NOMISS**

proc compare data = Adsl_dev compare = Adsl_val;
run;

The COMPARE Procedure
Comparison of WORK.ADSL_DEV with WORK.ADSL_VAL
(Method=EXACT)

Value level differences are clearly illustrated and so easy to identify

Illustrates there are more variables in ADSL_VAL

Observation Summary makes clear there are differences to investigate

Option to control numeric tolerance **CRITERION=**

Number of Observations with Some Compared Variables Unequal: 19.

compare_df

What is it? compare_df is a function in the compareDF package that allows you to compare your data on a line by line basis. It merges two data frames, with one or more grouping variables and creates a comparison of them. It is particularly useful when comparing smaller data frames. The comparison output shows all the rows for which at least one record has changed. If there are no differences of values across the two data frames, rows are not displayed. Below is an example of all available syntax, and output from a comparison of our two data frames highlighting the differences captured.

Selected Key Features:

- Control the number of observations in your compare
- Line by line comparison
- HTML Compare output is colour coded to illustrate differences between data frames
- Limit comparison to predetermined variables only **exclude=**
- Similar to using the **CRITERION=** option in Proc Compare there is the ability to set difference tolerance levels with the **tolerance=** argument
- Control the decimal precision in the output **round_output_to=**
- Comparison output can easily be created, use **create_output_table** function found in the compareDF package

syntax of available arguments for your compare

```
compare_df(  
  df_new,  
  df_old,  
  group_col,  
  exclude = NULL,  
  tolerance = 0,  
  tolerance_type = "ratio",  
  stop_on_error = TRUE,  
  keep_unchanged_rows = FALSE,  
  keep_unchanged_cols = TRUE,  
  round_output_to = 3  
)
```

optional syntax to create and control your output

```
create_output_table(  
  comparison_output,  
  output_type = ,  
  file_name = ,  
  limit = ,  
  color_scheme = ,  
  headers = NULL,  
  change_col_name = "chng_type",  
  group_col_name = )
```

Line by line comparison

Mismatches are easy to see and colour coded. The comparison is in a very different format to a Proc Compare

Option to control numeric tolerance **tolerance=**

comparedf

What is it? The comparedf function is found in the arsenal package in R. The function compares two data frames and reports any differences between them, much like using a Proc Compare. The function has good flexibility for controlling both the compare output and the tolerance level of differences in your output. The below syntax shows all available options in this function, the output illustrates which differences were captured.

Selected Key Features:

- Create clear and easy to read output
- Similarly to Proc Compare, limit how many differences are printed in the comparison
- Customise tolerance levels to detect change with multiple **tol.*=** options
- Detect differences by variable, and variables not common to both data frames
- Detect differences in the attributes of data frames and variables
- A detailed summary of the comparison is produced when used in conjunction with **summary** function, also in the arsenal package
- Additional utility functions in the arsenal package to help to validate data frames

syntax of all arguments and controls available for your compare

```
summary(comparedf(x, y, by = NULL, by.x = by, by.y = by,  
  tol.logical = "none",  
  tol.num = c("absolute", "percent", "pct"),  
  tol.num.val = sqrt(.Machine$double.eps),  
  int.as.num = FALSE,  
  tol.char = c("none", "trim", "case", "both"),  
  tol.factor = c("none", "levels", "labels"),  
  factor.as.char = FALSE,  
  tol.date = "absolute",  
  tol.date.val = 0,  
  tol.other = "none",  
  tol.vars = "none",  
  max.print.vars = NA,  
  max.print.obs = NA,  
  max.print.diffs.per.var = 10,  
  max.print.diffs = 50,  
  max.print.attrs = NA,  
  max.print.diff = 10 ))
```

Summary of overall comparison

statistic	value
Number of by-variables	1
Number of non-by variables in common	18
Number of variables compared	18
Number of variables in x but not y	0
Number of variables in y but not x	2
Number of variables compared with some values unequal	3
Number of variables compared with all values equal	15
Number of observations in common	25
Number of observations in x but not y	0
Number of observations in y but not x	0
Number of observations with some compared variables unequal	19
Number of observations with all compared variables equal	6
Number of values unequal	43

A detailed summary of the comparison using the **summary** function clearly shows there are differences to investigate

Value level differences are clearly illustrated and so easy to identify

Makes clear there are more variables in ADSL_VAL and lists the variable differences

Option to control numeric tolerance **tol.*=** options

diffdf

What is it? The diffdf package was developed specifically for the purpose of validating and comparing data frames, it contains functions for comparing data frames, alongside utility functions to outline differences and solve validation issues. The diffdf function in this package is itself designed to enable detailed comparison of two data frames, much like a Proc Compare. The below syntax shows all available options in this function and the output makes clear the differences that were captured.

Selected Key Features:

- Clearly output the number of differences found between variables and difference in observation counts between data frames
- It detects differences in attributes classes and column names
- Similar to using the **CRITERION=** option in a Proc Compare there is the ability to set tolerance level with the **tolerance=** argument
- Easily produce output that has familiarity for SAS programmers with BASE and COMPARE columns
- Helpful messages produced in the R console warning of differences detected
- Many other utility validation functions in the diffdf package

syntax of arguments for use

```
diffdf(  
  base,  
  compare,  
  keys = NULL,  
  suppress_warnings = FALSE,  
  strict_numeric = TRUE,  
  strict_factor = TRUE,  
  file = NULL,  
  tolerance = sqrt(.Machine$double.eps),  
  scale = NULL  
)
```

There are columns in COMPARE that are not in BASE !!
All rows are shown in table below

Useful messages in your R console clearly tells us that there are both variable and value level differences

Not all values Compared Equal!
All rows are shown in table below

Value level differences are clearly highlighted and so easy to identify

BASE and COMPARE columns provide Familiarity for SAS users

Option to control numeric tolerance **tolerance=**