Paper OS07

Use R to Track and Highlight the Changes Between Two Versions of Data to Have More Efficient Data Review

Disheng Huang, Sumitomo Pharma Oncology, Marlborough, USA
Zhonggai Li, Sumitomo Pharma Oncology, Marlborough, USA

ABSTRACT

Clinical data review on the subject level listings are usually required prior to each critical deliverables such as interim analysis, final analysis, etc. To help clinical team to perform a more efficient review, we have developed R codes to show only data that has been changed and highlight those changes (i.e. any new/changed records) between the current version with the previous version. This will not only reduce the reviewing time and resource dramatically, but also reduce the chance of overlooking changes.

INTRODUCTION

For ongoing phase 2/3 trials, performing the subject level data review is quite time consuming and error prone, as there may be couple hundreds of patients on the trials, and there may be thousands of data points entered or updated every day. In addition, when the critical database lock or snapshot is coming, reviewing the data may be needed on daily basis. Manually reviewing all the patient data for each data extraction can become a mission impossible. If there is a tool to help team identify what data has been newly entered or what data has been removed or modified, they could avoid the repeated data review on those un-changed data and focus their time on those changed ones. Thus, we developed R codes to programmatically identify those changes between different data snapshots.

This paper will demonstrate how to use R codes to build the track change files to show or highlight the data change as well as the challenges or lessons we learnt from this effort.

R-PACKAGES

The following R packages are utilized for building the track change files.

- dplyr: data manipulation and wrangling
- compareDF: compare and report the differences between two data.frames
- openxlsx: writing, and manipulating Microsoft Excel files

R PROGRAMMING

Here are the programming codes we used for comparing the current version of data with the previous version of data and generating the Microsoft Excel track change file.

Load the R packages:

```
library(dplyr)
library(openxlsx)
library(compareDF)
```

You need to provide two different version of data frames and the unique keys in group_col to make the comparison:

```
compare_res <- compareDF::compare_df(df_new, df_old, group_col,stop_on_error = FALSE)
```

Print out the difference between two data frames in R console:

```
print(compare_res$comparison_df)
```

Export the comparison output as Microsoft Excel report:

```
compare_report <- function(compare_result){
  comparison_df <- compare_result$comparison_df
  comparison_diff <- compare_result$comparison_table_diff
  wb <- createWorkbook()
  addWorksheet(wb, sheetName = "compare_report")
  writeData(wb, sheet = "compare_report", x = paste0("File Generation Date: ",
ymd(Sys.Date())),
            colNames = FALSE, startRow = 1)
  writeData(wb, sheet = "compare_report", x = comparison_df, colNames = TRUE, startRow
= 2)
  thinborder <- createStyle(border = "TopBottomLeftRight", borderColour = "black",
borderStyle = "thin")
  style_date <- createStyle(halign = "left", textDecoration = "bold", fontColour =
"#FFFFFF", fgFill = "#70ad47")
  redtxStyle <- createStyle(fontColour = "red", textDecoration = "bold", fgFill =
"yellow")
  greentxStyle <- createStyle(fontColour = "#007254", textDecoration = "bold", fgFill
= "yellow")
  style_boldwraptext <- createStyle(textDecoration = "bold", halign = "left", wrapText
= TRUE)
  addStyle(wb, sheet = "compare_report",style = thinborder, rows =
1:(nrow(comparison_df) + 2), cols = 1:ncol(comparison_df), stack = TRUE, gridExpand =
TRUE)
  addStyle(wb, sheet = "compare_report",style = style_date, rows = 1, cols =
1:ncol(comparison_df), stack = TRUE, gridExpand = TRUE)
  addStyle(wb, sheet = "compare_report", style = style_boldwraptext, rows = 2, cols =
1:ncol(comparison_df),stack = TRUE, gridExpand = TRUE)
  for (y in names(comparison_diff)[-which(names(comparison_diff) %in% c("grp",
"chng_type"))]){
    if(length(which(comparison_diff[[y]] %in% "-")) > 0){
      addStyle(wb, sheet = "compare_report", style = redtxStyle,
               rows = (which(comparison_diff[[y]] %in% "-") + 2), cols =
which(names(comparison_df) %in% y),
               stack = TRUE, gridExpand = TRUE)
    }

    if(length(which(comparison_diff[[y]] %in% "+")) > 0){
      addStyle(wb, sheet = "compare_report", style = greentxStyle,
               rows = (which(comparison_diff[[y]] %in% "+") + 2), cols =
which(names(comparison_df) %in% y),
               stack = TRUE, gridExpand = TRUE)
    }
  }
  saveWorkbook(wb, "compare_report.xlsx",overwrite = TRUE)
}

compare_report(compare_result = compare_res)
```

EXAMPLES AND OUTPUTS

This section will give you some examples about how to use the programming codes we provided and what are the outputs look like.

CASE 1: RECORD ADDED

Based on the sample codes below, you may find the only difference between those two data frames is there is one additional record added in the new data frame (df2). This new record could be quickly found by using the R function provided.

```
df1 <- data.frame(SUBJID = paste0("A", 1:3), SEX = c("F","M","F"), AGE = c(21, 56,
34), stringsAsFactors = FALSE)
df2 <- data.frame(SUBJID = paste0("A", 1:4), SEX = c("F","M","F","F"), AGE = c(21, 56,
34, 67), stringsAsFactors = FALSE)
compare_res <- compareDF::compare_df(df_new = df2, df_old = df1, group_col = "SUBJID",
stop_on_error = FALSE)
compare_report(compare_result = compare_res)
```

The screenshot below shows there is a new record added in the new data frame. The font is colored in green and the background is highlighted in yellow.

File Generation Date: 2023-02-12			
SUBJID	chng_type	SEX	AGE
A4	+	F	67

CASE 2: RECORD DELETED

Based on the sample codes below, you may find the only difference between those two data frames is there is one record deleted in the new data frame (df2). This could be found by using the R function provided.

```
df1 <- data.frame(SUBJID = paste0("A", 1:3), SEX = c("F","M","F"), AGE = c(21, 56,
34), stringsAsFactors = FALSE)
df2 <- data.frame(SUBJID = paste0("A", 1:2), SEX = c("F","M"), AGE = c(21, 56),
stringsAsFactors = FALSE)
compare_res <- compareDF::compare_df(df_new = df2, df_old = df1, group_col = "SUBJID",
stop_on_error = FALSE)
compare_report(compare_result = compare_res)
```

The screenshot below shows there is one record removed from the new data frame. The font is colored in red and the background is highlighted in yellow.

File Generation Date: 2023-02-12			
SUBJID	chng_type	SEX	AGE
A3	-	F	34

CASE 3: RECORD CHANGED

Based on the sample codes below, you may find the SEX for SUBJID A1 is changed from “F” to “M” and the age for A3 is changed from “34” to “43. These changes could be found by using the R function provided.

```
df1 <- data.frame(SUBJID = paste0("A", 1:3), SEX = c("F","M","F"), AGE = c(21, 56,
34), stringsAsFactors = FALSE)
df2 <- data.frame(SUBJID = paste0("A", 1:3), SEX = c("M","M","F"), AGE = c(21, 56,
43), stringsAsFactors = FALSE)
compare_res <- compareDF::compare_df(df_new = df2, df_old = df1, group_col = "SUBJID",
stop_on_error = FALSE)
compare_report(compare_result = compare_res)
```

Below please find the comparison result between those two data frames. The fonts for the new records are colored in green and the fonts for the old records are colored in red. The backgrounds for those changes are highlighted in yellow.

File Generation Date: 2023-02-14			
SUBJID	chng_type	SEX	AGE
A1	+	M	21
A1	-	F	21
A3	+	F	43
A3	-	F	34

APPLICATIONS

We have applied this tool to create the track changes files for patient profile, data cleaning listings and efficacy smart listings. Those track change files are intuitive and easy to review.

CHALLENGES AND LIMITATIONS

There are two major challenges about using this method to create the track change files.

The function `compareDF::compare_df` needs to know the unique keys to compare the data sets. The unique keys should be the same in both data sets. However, in most of the cases, there would be 60 ~ 80 case report forms or data sets in one study. Thus, when we initially develop the track change files, we may need to check each of those forms to identify the unique keys in each form.

The function `compareDF::compare_df` can only compare the two data sets with the same number of columns and same column names. It is very common that we would have the case report form update or database migration due to the protocol amendment. Thus, the structure of the data sets would be different from time to time. Some new columns would be added. Some old columns would be retired or modified. This function or method could not detect those changes, and it would report these cases as errors and stop the comparison.

CONCLUSION

Although there are some challenges and limitations that we may need to pay attention to, using R to build the track change files to highlight the changes between two version of data sets is very convenient. This method can reduce the clinical data reviewing time dramatically as well as improve the quality for data review. Overall, this is a very useful tool that can be applied to all clinical data review especially for the subject level data review in phase 2/3 trials.

REFERENCES

Links related to R packages:

- https://dplyr.tidyverse.org/
- https://cran.r-project.org/web/packages/openxlsx/openxlsx.pdf
- https://rpubs.com/alexsanjoseph/compareDF_intro

ACKNOWLEDGMENTS

Aaron Chen, Sylvia Chen

CONTACT INFORMATION

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

Author Name: Disheng Huang
Company: Sumitomo Pharma Oncology
Address: 84 Waterford Dr
City / Postcode: Marlborough, MA 01752
Email: disheng.huang@oncology.sumitomo-pharma.com
LinkedIn: https://www.linkedin.com/in/disheng-huang/

Author Name: Zhonggai Li
Company: Sumitomo Pharma Oncology
Address: 84 Waterford Dr
City / Postcode: Marlborough, MA 01752
Email: kevin.li@oncology.sumitomo-pharma.com