PharmaSUG China 2023 - Paper CC-140

Maximize Efficiency: Batch Compare Files with Ease Using Beyond Compare

Gary Jia and Sean Yan, ICON

ABSTRACT

It is quite common to compare files in our daily work. The most used method is to compare datasets with proc compare, but it is also important for other types of files. It is common to compare (.txt), metadata (.xlsx), or TFL (.rtf) between different versions, and sometimes you will check whether or not the files are identical. If you have too many files to compare, and you can't guarantee the quality just by looking at them. A useful tool and its application in batch comparison of documents are introduced.

Beyond Compare is a tool for comparing files and folders. It offers a user-friendly interface, supports a wide variety of file formats, and can be customized according to specific requirements. Despite the convenience of the tool's user interface, batch comparison is more useful in most situations. Fortunately, Beyond Compare allows users to write a script file to compare without opening software. Once the method and the file address are written into the script file, the command can be executed in the CMD, and the result of the comparison will be output to the indicated path. Running the batch command code in CMD for batch comparison is very helpful. Using R, we can write a CMD command into a .bat file with its loop logic, which saves a great deal of effort. You can also view any comparison results that appear in the style you preconfigure in the script file.

In this article, it is recommended that we use tools to do this meticulous work instead of using our eyes to guarantee the quality of the work. Because the comparison results are obvious, it makes it easier for the programmer to distinguish between two files, not just by the type of file or the number of files.

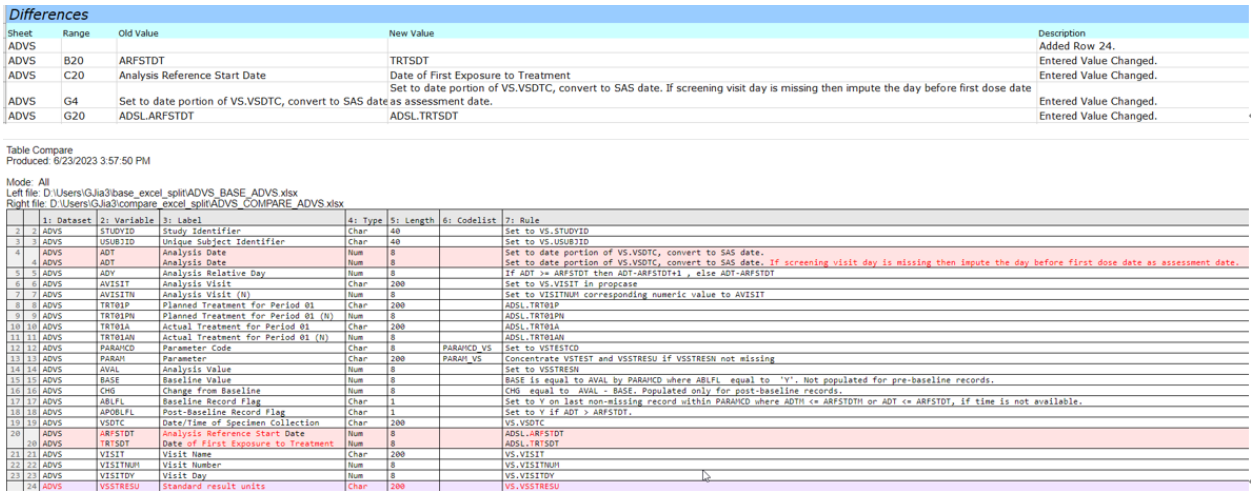
INTRODUCTION

In our work, we often compare files, with the most common being comparing datasets. SAS has a built-in feature called "proc compare" which is very useful and can quickly identify differences. However, for comparing other types of files, such as two study analysis contents that are similar and use the same database, it is necessary to compare and identify the differences between them. It can also be useful for tracking updates between versions of code or verifying if files in two folders match, and identifying the differences if they don't. These comparison tasks consume a significant amount of our time. To improve efficiency and accuracy, this paper introduces the use of Beyond Compare to easily compare different types of files, and even efficiently perform batch comparisons.

Some may wonder if other software can also perform these compare functions, so what are the advantages of Beyond Compare?

- Beyond Compare supports various file comparisons. It can simultaneously compare different types of files, including SAS and R code.
- The comparison results in Beyond Compare can be customized in terms of output style. You can view them in the context or choose to display only the differences, which are usually easily noticeable and quickly identifiable.
- Beyond Compare supports running CMD commands for background calls. This enables batch file comparison and outputting of compare results through CMD commands, significantly improving efficiency for comparing a large number of files.

In summary, Beyond Compare is a very useful comparison tool that enhances our efficiency, especially when working with batch file comparisons.



Differences

Sheet	Range	Old Value	New Value	Description
ADVS				Added Row 24.
ADVS	B20	ARFSTDT	TRTSDT	Entered Value Changed.
ADVS	C20	Analysis Reference Start Date	Date of First Exposure to Treatment	Entered Value Changed.
ADVS	G4	Set to date portion of VS.VSDTC, convert to SAS date as assessment date.	Set to date portion of VS.VSDTC, convert to SAS date. If screening visit day is missing then impute the day before first dose date	Entered Value Changed.
ADVS	G20	ADSL.ARFSTDT	ADSL.TRTSDT	Entered Value Changed.

Table Compare
Produced: 6/23/2023 3:57:50 PM

Mode: All
Left file: D:\Users\GJia3\base_excel_split\ADVS_BASE_ADVS.xlsx
Right file: D:\Users\GJia3\compare_excel_split\ADVS_COMPARE_ADVS.xlsx

		1: Dataset	2: Variable	3: Label	4: Type	5: Length	6: Codelist	7: Rule
2	2	ADVS	STUDYID	Study Identifier	Char	40		Set to VS.STUDYID
3	3	ADVS	USUBJID	Unique Subject Identifier	Char	40		Set to VS.USUBJID
4		ADVS	ADT	Analysis Date	Num	8		Set to date portion of VS.VSDTC, convert to SAS date.
	4	ADVS	ADT	Analysis Date	Num	8		Set to date portion of VS.VSDTC, convert to SAS date. If screening visit day is missing then impute the day before first dose date as assessment date.
5	5	ADVS	ADY	Analysis Relative Day	Num	8		If ADT >= ARFSTDT then ADT-ARFSTDT+1 , else ADT-ARFSTDT
6	6	ADVS	AVISIT	Analysis Visit	Char	200		Set to VS.VISIT in propcase
7	7	ADVS	AVISITN	Analysis Visit (N)	Num	8		Set to VISITNUM corresponding numeric value to AVISIT
8	8	ADVS	TRT01P	Planned Treatment for Period 01	Char	200		ADSL.TRT01P
9	9	ADVS	TRT01PN	Planned Treatment for Period 01 (N)	Num	8		ADSL.TRT01PN
10	10	ADVS	TRT01A	Actual Treatment for Period 01	Char	200		ADSL.TRT01A
11	11	ADVS	TRT01AN	Actual Treatment for Period 01 (N)	Num	8		ADSL.TRT01AN
12	12	ADVS	PARAMCD	Parameter Code	Char	8	PARAMCD_VS	Set to VSTESTCD
13	13	ADVS	PARAM	Parameter	Char	200	PARAM_VS	Concentrate VSTEST and VSSTRESU if VSSTRESN not missing
14	14	ADVS	AVAL	Analysis Value	Num	8		Set to VSSTRESN
15	15	ADVS	BASE	Baseline Value	Num	8		BASE is equal to AVAL by PARAMCD where ABLFL equal to 'Y'. Not populated for pre-baseline records.
16	16	ADVS	CHG	Change from Baseline	Num	8		CHG equal to AVAL - BASE. Populated only for post-baseline records.
17	17	ADVS	ABLFL	Baseline Record Flag	Char	1		Set to Y on last non-missing record within PARAMCD where ADTH <= ARFSTDTH or ADT <= ARFSTDT, if time is not available.
18	18	ADVS	APOBLFL	Post-Baseline Record Flag	Char	1		Set to Y if ADT > ARFSTDT.
19	19	ADVS	VSDTC	Date/Time of Specimen Collection	Char	200		VS.VSDTC
20		ADVS	ARFSTDT	Analysis Reference Start Date	Num	8		ADSL.ARFSTDT
	20	ADVS	TRTSDT	Date of First Exposure to Treatment	Num	8		ADSL.TRTSDT
21	21	ADVS	VISIT	Visit Name	Char	200		VS.VISIT
22	22	ADVS	VISITNUM	Visit Number	Num	8		VS.VISITNUM
23	23	ADVS	VISITDY	Visit Day	Num	8		VS.VISITDY
	24	ADVS	VSSTRESU	Standard result units	Char	200		VS.VSSTRESU

Display 1. Comparison of Spreadsheet Compare(Upper Half) and Beyond Compare(Lower Half) Results

COMMAND LINE AND SCRIPTING REFERENCE

COMMAND LINE REFERENCE

Beyond Compare supports the Command Line, which is crucial for batch comparing files. You can refer to the command line reference in the Beyond Compare help contents to learn more. Common commands are listed in Table 1 below.

Purpose	Usage
Pair of folders	Opens a new Folder Compare view with the specified base folders. For example: BCompare.exe "C:\Left Folder" "C:\Right Folder"
Pair of files	Opens the specified files in the associated file view. For example: BCompare.exe "C:\Left File.ext" "C:\Right File.ext"
3 files	Opens a Text Merge view with the specified files in the left, right, and center panes. For example: BCompare.exe C:\Left.ext C:\Right.ext C:\Center.ext
Script file	Automatically executes a list of commands without using a view. For example: BCompare.exe "@C:\My Script.txt"

Table 1. Command Line Reference

SCRIPTING REFERENCE

Since the research method in this paper involves invoking Beyond Compare in the background and batch outputting results without opening the user interface, we only need to use the method of calling a Script file(See Table 2). This method automatically executes a list of commands without using a view. In simple terms, a Script file is used to define the paths of the files you want to compare, the comparison rules and settings, and the output settings for the comparison results, following the syntax provided in the Scripting Reference in the help contents.

Therefore, the core idea to solve the problem is to batch write the command:

```
"BCompare.exe /silent @"your_script.txt" "file1" "file2" "compare_result.html"
```

to achieve the purpose of batch comparing files and outputting the comparison results.

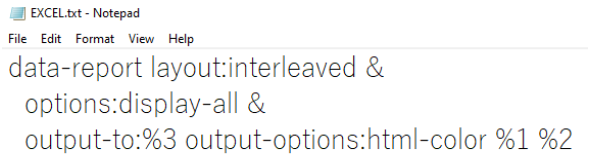
Grammar	Usage
DATA-REPORT	Generates a Table Compare report of the currently selected files.
FILE-REPORT	Generates a report of the currently selected files based on the type of files processed. For example, it will produce a DATA-REPORT for file types associated with a Table Compare file format.
FOLDER-REPORT	Generates a folder comparison report of the currently loaded base folders. Folders are shown in their current state. To include subfolders add an expand all command before this.
TEXT-REPORT	Generates a Text Compare report of the currently selected files.

Table 2. Scripting Reference

BATCH COMPARES EXCEL WITH R

DEFINE EXCEL SCRIPT FILE

Before comparing EXCEL files, it is necessary to define a script file. We use a data-report layout to perform table comparisons and display the comparison results in an interleaved manner, showing all the differences. The comparison results are output in HTML format with color-coded markings.



Display 2. Excel Script

CHOOSING R OR SAS

Once the script file is prepared, we need to use R's loop to write the compare command into a .bat file. Here, I will explain why we are not using a more familiar language like SAS for this process. Both SAS and R can handle the looping logic, but SAS has a maximum length limitation on text output (even with the option linesize=max). This maximum length varies depending on the operating system and version, but typically it is 256 characters. Since our command contains at least three paths, it is easy to exceed the 256-character limit and cause line breaks when outputting the command. This will result in CMD not being able to recognize the compare command properly. On the other hand, R does not have this limitation, so choosing a non-SAS language is more reliable.

CODE INTRODUCTION

Split excel files

Since Beyond Compare only compares the default sheet when comparing EXCEL files with multiple sheets, and we need to compare each sheet, we first need to save each EXCEL file as EXCEL_SHEETNAME.xlsx. These derived files can be deleted after the comparison. Here is a sample reference code:

```
#Initialization
library(stringr)
library(readxl)
library(writexl)
#Split EXCEL function(should input two compared folder path and output result path)
split <- function(base_path, compare_path ,results_path){
  base_sheets <- excel_sheets(base_path)
  #Derive each base EXCEL_Sheetname into results_path/base_excel_split
  base_split_dir <<- paste0(results_path,"/base_excel_split")
  comp_sheets <- excel_sheets(compare_path)
```

```
  #Derive each compare EXCEL_Sheetname into results_path/compare_excel_split
  comp_split_dir <- paste0(results_path,"/compare_excel_split")

  #create a sub-folder if it doesn't exist
  if (!file.exists(base_split_dir)) {
    dir.create(base_split_dir, recursive = TRUE)
  }
  if (!file.exists(comp_split_dir)) {
    dir.create(comp_split_dir, recursive = TRUE)
  }

  #Loop each sheet and save it.
  for (sheet_name in base_sheets) {
    data <- read_excel(base_path, sheet = sheet_name)
    output_file <<- paste0(base_split_dir,"/", str_extract(base_path,
"(?<=\\/)[^\\/]+(?=\\.xlsx$)"),"_",str_replace_all(sheet_name," ","_"), ".xlsx")
    write_xlsx(data, output_file)
  }

  for (sheet_name in comp_sheets) {
    data <- read_excel(compare_path, sheet = sheet_name)
    output_file <<- paste0(comp_split_dir,"/", str_extract(compare_path,
"(?<=\\/)[^\\/]+(?=\\.xlsx$)"),"_" ,str_replace_all(sheet_name," ","_"), ".xlsx")
    write_xlsx(data, output_file)
    comp_list <<- c(comp_list,output_file)
  }
}
#split("PATH1/EXCELA.xlsx" , " PATH2/EXCELB.xlsx " , "OUTPUT_PATH")
```

Write command into .bat file

After completing the split of EXCEL files, we can write the compare command to the bat file. Please refer to the following example command:

```
#Initialization
output_bat <- "Your_Path/output.bat"
text <- c()
#Define batch_text function to write command into .bat file
batch_text <- function(base_path, compare_path ,results_path){
  text <<- c(text,paste0('BCompare.exe /silent @"Your_Path/excel_script.txt" ' ,'"', base_path,
'" ' , '"', compare_path, '" ' , '"', results_path ,'"'))
}
#batch_text("OUTPUT_PATH/base_excel_split/EXCELA_SHEET1.xlsx" , " OUTPUT_PATH
/compare_excel_split/EXCELA_SHEET1.xlsx " , "OUTPUT_PATH/EXCELA_SHEET1.html")

writeLines(text, output_bat)
#delete split folder.
unlink(paste0(result_path,"/base_excel_split"), recursive = TRUE)
unlink(paste0(result_path,"/compare_excel_split"), recursive = TRUE)
```

CASE 1

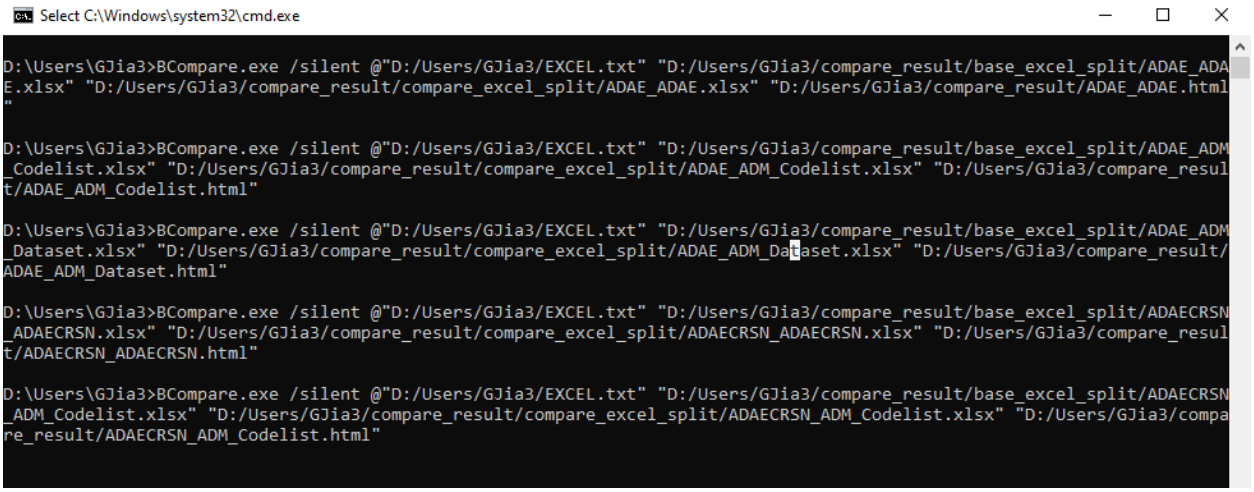
Since the cases may vary, I will only provide an example used in my study. Currently, my project group is very similar to another project, so I can refer to the metadata from the mature project group. However, mistakes in the database or the reference project may cause differences in the metadata. I need to examine the deviations and standardize them. Here is a sample reference code:

```
base_path <- "D:/Users/GJia3/Reference_study"
comp_path <- "D:/Users/GJia3/My_study"
result_path <- "D:/Users/GJia3/compare_result"
file_list <- list.files(base_path, pattern = "*.xlsx", full.names = FALSE)
for (file in file_list) {
  split(paste0(base_path,"/",file) , paste0(comp_path,"/",file) , result_path)
}
split_file_list <- list.files(paste0(result_path,"/base_excel_split"), pattern = "*.xlsx",
full.names = FALSE)
for (file in split_file_list) {
  batch_text(paste0(result_path,"/base_excel_split/",file) ,
paste0(result_path,"/compare_excel_split/",file) , paste0(result_path,"/",strsplit(file,
"\\.xlsx")[[1]],".html"))
```

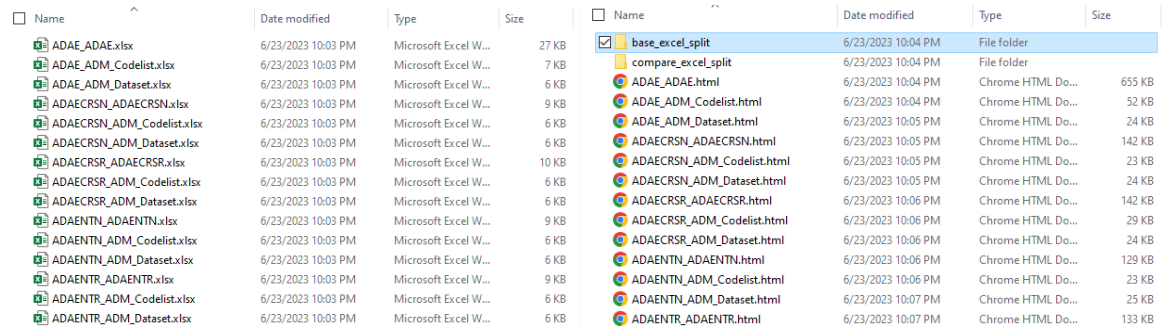
```
}
writeLines(text, output_bat)

#delete split folder.
unlink(paste0(result_path,"/base_excel_split"), recursive = TRUE)
unlink(paste0(result_path,"/compare_excel_split"), recursive = TRUE)
```

All Excel files(.xlsx) located in “Reference_study” and “My_study” will be split into “EXCEL_SHEETNAME.xlsx”(See Display 4 left half). When you run “output.bat” and wait a few moments(See Display 3), All compare results will be generated. Now you can delete the derived split Excel files and review your results of the comparison.



Display 3. Unleashing the Power of Beyond Compare by using CMD

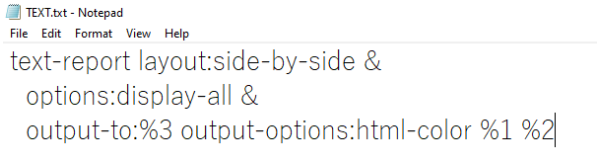


Display 4. Split Excel screenshot(Left half) and the results of the comparison(Right half)

BATCH COMPARES TEXT WITH R

DEFINE TEXT SCRIPT FILE

Just like defining an EXCEL script file, we also need to define a text script file to instruct Beyond Compare to perform text comparison. Unlike the previous EXCEL script file, this time I will present the comparison results in a side-by-side layout.



Display 5. Text Script

CASE 2

In our work, we may encounter situations where we have some standard macros for reference. However, there might be a time when we make modifications to the macros for a specific study but fail to keep any records of the changes. After some time passes, we may struggle to remember what modifications were made. In such cases, batch compare code becomes particularly important. Here is a reference code for this case:

```
#Initialization
library(stringr)
library(readxl)
library(writexl)
output_bat <- "D:/Users/GJia3/output.bat"
text <- c()
#batch compare text(code)
batch_text <- function(base_path, compare_path ,results_path){
  text <<- c(text,paste0('BCompare.exe /silent @"D:/Users/GJia3/TEXT.txt" ' ,'"', base_path,
'" ' , '"', compare_path, '" ' , '"', results_path ,'"'))
}
base_path <- "D:/Users/GJia3/Code_version1"
comp_path <- "D:/Users/GJia3/Code_version2"
result_path <- "D:/Users/GJia3/compare_result"
code_list <- list.files(base_path, pattern = "*.sas", full.names = FALSE)
for (file in code_list) {
  batch_text(paste0(base_path,"/",file) , paste0(comp_path,"/",file) ,
paste0(result_path,"/",strsplit(file, "\\.sas")[[1]],".html"))
}
writeLines(text, output_bat)
```

All compare results will be generated. We can open each result file to see the results.

Display 6. Text side-by-side comparison result

CONCLUSION

If you often suffer from headaches when comparing files and these tasks require a significant amount of time and effort, it is better to entrust them to Beyond Compare. Combining R language batch compare files, can greatly improve your work efficiency, and the results obtained are more accurate than reviewing comparisons.

REFERENCES

"html 批量选择文件类型,Beyond Compare 脚本：命令行批量比较文件并生成 html 格式的差异报告" June 7, 2021. https://blog.csdn.net/weixin_30366435/article/details/117861341

"Beyond Compare help" https://www.scootersoftware.com./v4help/index.html

ACKNOWLEDGMENTS

We would specifically like to acknowledge Sean Yan, Tina Liu(Janssen), and Li Chen(Janssen) for their assistance and for offering great inspiration and feedback.

CONTACT INFORMATION

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

Gary Jia, ICON
Zhixiang.jia@iconplc.com

Sean Yan, ICON
Yan.Sean@iconplc.com