PharmaSUG China 2025 – Paper PO-103

A R function to quickly batch check the QC result in one project with hybrid programming languages

Perphy ZHAO, Sanofi;
Michael WANG, Sanofi

Abstract

Double programming is highly important and necessary for clinical data analysis. This paper aims to introduce a customed R function to quickly batch check the QC result for data or output when hybrid programming languages such R and SAS are used in a project, which not only can improve the effectiveness, but also has the easy and strict mode to choose.

Introduction

Statistical analysis of clinical trial data is usually double program, which refers to two independent programmers' programming, and then comparing and validating the results. Its importance lies in ensuring the accuracy and reliability of analytical results to eliminate human errors and enhance the scientific validity and credibility of data analysis and helps identify and rectify discrepancies, thereby minimizing the risk of false conclusions. Therefore, whatever tools used to the analysis, it should and must be double programming to ensure the statistical quality. As a shared, ideal and popular programming language in recent years, R has great strengths in statistical analysis, data wrangling and visualization, and more and more programmers in pharmaceutical industry are moving from SAS to R programming. In order to help ones to know about comparison status in a project, we create a customed function to batch check the comparison status of data or output quickly.

Function illustration

Package used

```
library(openxlsx)
library(xlsx)
library(rtf)
library(tidyverse)
library(diffdf)
library(readtext)
```

Prepare the reflist or the qc plan

For reflist, the following columns must be contained in reflist.

filename	title	pgmname	pgm_r	qcpgmname	qcpgm_r
totally_pass	table 1	totally_pass	Y	v_totally_pass	Y
partial_pass	table 2	partial_pass	Y	v_partial_pass	Y
totally_nopass	table 3	totally_nopass	N	v_totally_nopass	N
adeg	adeg	adeg	N	v_adeg	N
adex	adex	adex	N	v_adex	N
adlb	adlb	adlb	N	v_adlb	N
ae_aesi_socpt_s_t	ae_aesi_socpt_s_t	ae_aesi_socpt_s_t	N	v_ae_aesi_socpt_s_t	N
osa_ecg_pcsa_def_s_l	osa_ecg_pcsa_def_s_l	osa_ecg_pcsa_def_s_l	N	v_osa_ecg_pcsa_def_s_l	N
ades	ades	ades	N	v_ades	N

filename: output name, must be unique

title: title for each output, must be unique

pgmname: leading program name

pgm_r: indicate whether the leading program is created by R

qcpgmname: QC program name

qcpgm_r: indicate whether the QC program is created by R

Individual QC program

For individual TLG, QCer should create rtf, txt or other format file:

For SAS, use the ods statement to wrap the compare procedure and output the comparison result to the rtf, txt or other format file.

```
ods listing close;
ods rtf file="&W_ROOT./&W_MODWORK.OPS/&w_compound/&w_study/&w_analysis/QC/OUTPUT/v_&chkoutput..rtf";

proc compare base=base comp=comp out=_dif outbase outcomp outdif outnoeq;
run;

ods rtf close;
ods listing;
```

For R: use sink() function to wrap the diffdf() function to capture the information in console and create rtf, txt or other format file.

```
filename <- 'filepath/v_partial_pass.txt'
sink(filename)
diffdf(cars,cars2)
sink()
```

Batch check function

The nested function:

1. read_qc_file: Read the QC result file.

```
read_qc_file <<- function(x=NULL){
  if ( file.exists(file.path(QCO, paste0('v_',trimws(x),'.rtf')))){
    x_temp <-readtext(file.path(QCO, paste0('v_',trimws(x),'.rtf')))
  }else if (file.exists(file.path(QCO, paste0('v_',trimws(x),'.txt')))){
    x_temp <- readtext(file.path(QCO, paste0('v_',trimws(x),'.txt')))
  }else if (file.exists(file.path(QCO, paste0('v_',trimws(x),'.doc')))){
    x_temp <- readtext(file.path(QCO, paste0('v_',trimws(x),'.doc')))
  }else if (file.exists(file.path(QCO, paste0('v_',trimws(x),'.docx')))){
    x_temp <- readtext(file.path(QCO, paste0('v_',trimws(x),'.docx')))
  }else if (file.exists(file.path(QCO, paste0('v_',trimws(x),'.pdf')))){
    x_temp <- readtext(file.path(QCO, paste0('v_',trimws(x),'.pdf')))
  }else {
    x_temp <- tibble(doc_id=c(''),text=c(''))
  }
  return(x_temp)
}
```

Use this function to read all the QC file in the reflist by apply() function.

```
read_qc_file_step <- reflist  %>%
  mutate(
    check=lapply(filename, read_qc_file)
  ) %>%
  unnest_wider(check)
```

2. get_info: Get the information of programs or files.

```
get_info <- function(x=NULL ,filepath=NULL){

  file_info <- file.info(file.path(filepath, trimws(x)))
  return(file_info)
}
```

Use this function to get the information of leading program, output and QC program and output information such the modification datetime in the reflist or QC plan by apply() function.

```
pgm_file_info <-read_qc_file_step %>%
  mutate(
    pgmname_temp=if_else(
      pgm_r=='Y',
      paste0(pgmname,'.R'),
      paste0(pgmname,'.sas'),
    ),
    info_pgm=lapply(pgmname_temp, get_info,filepath=REPP),

    file_temp=paste0(filename,'_x.rtf'),
    info_file=lapply(file_temp, get_info,filepath=REPO),

    qcpgmname_temp=if_else(
      qcpgm_r=='Y',
      paste0(qcpgmname,'.R'),
      paste0(qcpgmname,'.sas'),
    ),
    info_qcpgm=lapply(qcpgmname_temp, get_info,filepath=QCP),

    qcfile_temp=doc_id,
    info_qcfile=lapply(qcfile_temp, get_info,filepath=QCO),
    qcfile=gsub('(.txt)|(.rtf)','',qcfile_temp),
    qcfile1=if_else(
      qcfile_temp!='' | ! is.na(qcfile_temp),
      paste0(QCO,'/',qcfile_temp),
      ''
    )
  ) %>%
  unnest_wider(col=c(info_pgm,info_file,info_qcpgm,info_qcfile),names_sep = '_')
```

Capture the key information by regular expression from the QC file, such as *'no issues were found', 'there are columns in base and compare with differing attributes', 'there are rows in base that are not in compare', 'there are rows in compare that are not in base', 'there are columns in compare that are not in base', 'there are columns in base that are not in compare', 'not all values compared equal'* and other key information to check whether the data or the rtf pass QC or not. The comparison result has the following information:

'Both attributes and value pass'

'Attributes not pass but value pass'

'Attributes pass but value not pass'

'Both attributes and value not pass'

'Both attributes and value pass, but QC-time<=Source-time, Please re-run'

'Attributes not pass but value pass, but QC-time<=Source-time, Please re-run'

'Attributes pass but value not pass, but QC-time<=Source-time, Please re-run'

'Both attributes and value not pass, but QC-time<=Source-time, Please re-run'

'Not compare'

Compare the datetime of program and data/output for lead and QC programmer, lead and QC data/output, and check whether the program need to rerun after modification. The status of lead and QC program has the following information:

'Not start'

'Programming'

'Not meet time logic, Please re-run'

'Ready for compare'

The final excel file contains additional columns as below

info_pgm_mtime	info_file_mtime	info_qcpgm_mtime	info_qcfile_mtime	qcfile	program_status	qc program status	compare status
2025-02-26 14:53:02	2025-02-26 15:53:02	2025-03-03 10:53:02	2025-03-05 14:44:02	v_totally_pass	Ready for compare	Ready for compare	Both attributes and value pass
2025-02-26 14:53:02	2025-02-26 15:53:02	2025-03-03 10:53:02	2025-03-05 14:45:02	v_partial_pass	Ready for compare	Ready for compare	Attributes not pass but value pass
2025-02-26 14:53:02	2025-02-26 15:53:02	2025-03-03 10:53:02	2025-03-05 14:46:02	v_totally_nopass	Ready for compare	Ready for compare	Both attributes and value not pass
2025-02-26 15:17:54	2025-02-26 14:17:54	2025-03-03 10:17:54	2025-03-05 15:17:54	v_adeg	Not meet time logic, Please re-run	Ready for compare	Attributes not pass but value pass
2025-02-26 15:15:34	2025-02-26 16:15:34	2025-03-03 10:15:34	2025-03-05 15:15:34	v_adex	Ready for compare	Ready for compare	Attributes not pass but value pass
2025-02-26 15:02:42	2025-02-26 16:02:42	2025-03-03 10:02:42	2025-02-26 09:30:42	v_adlb	Ready for compare	Not meet time logic, Please re-run	Attributes not pass but value pass, but QC-time<=Source-time, Please re-run
		2025-03-03 10:02:16			Not start	Programming	Not compare
2025-02-26 15:10:54		2025-03-03 10:10:54			Programming	Programming	Not compare
					Not start	Not start	Not compare

The first four columns are the leading program datetime, leading data/output datetime, QC program datetime, QC output datetime separately.

User can quickly check the status of leading program, QC program and comparison status from last three columns. It's easy comparison mode when the last column contains **'value pass'** and does not contain **'but QC-time<=Source-time, Please re-run'**. Otherwise, it's strict comparison mode when the last column in **'Both attributes and value pass'** and does not contain **'but QC-time<=Source-time, Please re-run'**. Users can also check the difference from qcfile column, which is hyperlink and click it, then the file will open.

Function parameters illustration

```
validation_tlg (
    reflistname        = 'reflist.xlsx', # reflist name
    pgmpath            = REPP,           # leading program path
    datapath           = REPD            # leading data path
    filepath           = REPO,           # leading file path
    qcpgmpath          = QCP,            # QC program path
    qcfilepath         = QCO,            # QC file path
    validationfilepath = QCO,            # validation file path
    validationfile     = 'validation'    # validation file name
             )
```

Potential error

R is case-sensitive, please pay attention to the columns' name when create reflist.

Reference

[1] Pharmasug-China-2024-CC10010_Final_Paper.pdf

ACKNOWLEGEMENTS

The authors would like to take this opportunity to thank Sanofi and my team for facilitating and supporting this paper.

CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Contact the authors at:

Perphy ZHAO
perphyzhao@163.com

Michael WANG
michael3.wang@sanofi.com