PHUSE EU Connect 2021

R programming for Validation

Kentaro Arai,
Novartis Pharma K.K.

NOVARTIS | Reimagining Medicine

Disclaimer

- All information provided in this slides is provided for information purposes only
- The views in this presentation are those of the speaker and not necessarily of Novartis.

Agenda

- Introduction
- ADaM validation
- TFLs validation
- Key takeaways

NOVARTIS | Reimagining Medicine

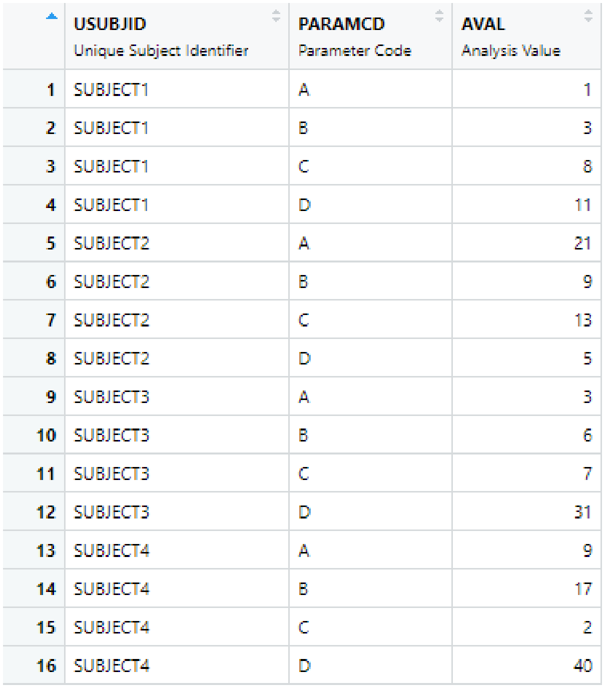
Introduction

- R proposes thousands of implementations for statistical analysis or data analysis
- Free software
- There are millions of R users worldwide
- There are many materials to learn R
- R performances are similar or even better than most of the commercial softwares
- Many Pharmaceutical companies have started to use R

ADaM validation

- Data review by R
- Example of R programming for data creation
- Key differences between SAS and R
- Compare logic

Data review by R: Outline



	USUBJID Unique Subject Identifier	PARAMCD Parameter Code	AVAL Analysis Value
1	SUBJECT1	A	1
2	SUBJECT1	B	3
3	SUBJECT1	C	8
4	SUBJECT1	D	11
5	SUBJECT2	A	21
6	SUBJECT2	B	9
7	SUBJECT2	C	13
8	SUBJECT2	D	5
9	SUBJECT3	A	3
10	SUBJECT3	B	6
11	SUBJECT3	C	7
12	SUBJECT3	D	31
13	SUBJECT4	A	9
14	SUBJECT4	B	17
15	SUBJECT4	C	2
16	SUBJECT4	D	40

Introduce the simple logic for data review using this test data

- How to show the data
- How to filter the data
- How to summarize the data

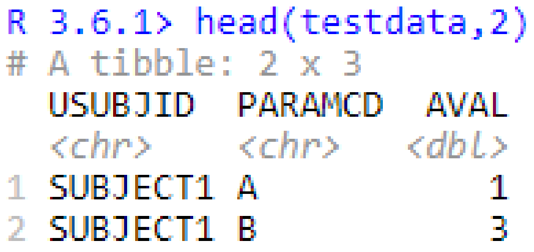
Data Review by R: select the lines

#Display first 6 lines of data set x

head(x)

#Display first n lines of data set x

head(x, n)



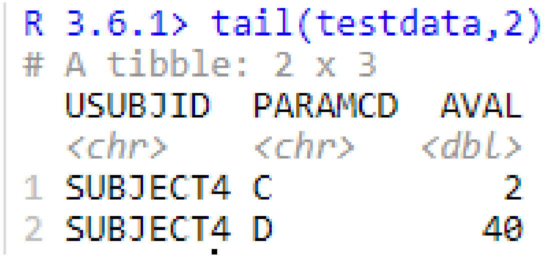
```
R 3.6.1> head(testdata,2)
# A tibble: 2 x 3
  USUBJID  PARAMCD  AVAL
  <chr>    <chr>   <dbl>
1 SUBJECT1 A           1
2 SUBJECT1 B           3
```

#Display last 6 lines of data set x

tail(x)

#Display last n lines of data set x

tail(x, n)

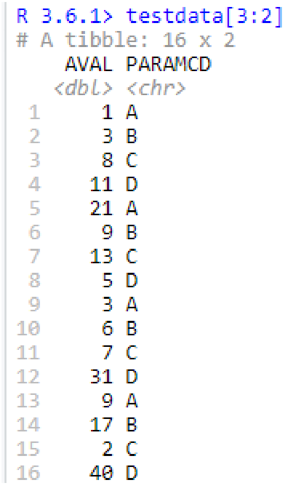


```
R 3.6.1> tail(testdata,2)
# A tibble: 2 x 3
  USUBJID  PARAMCD  AVAL
  <chr>    <chr>   <dbl>
1 SUBJECT4 C           2
2 SUBJECT4 D          40
```

NOVARTIS | Reimagining Medicine

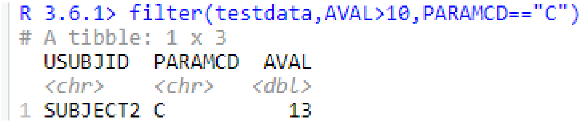
Data Review by R : Filter the data

```
# Select 2nd row
testdata[2]

# Select multiple rows, 3rd and 2nd row
testdata[3:2]
```



```
R 3.6.1> testdata[3:2]
# A tibble: 16 x 2
    AVAL PARAMCD
   <dbl> <chr>
 1     1 A
 2     3 B
 3     8 C
 4    11 D
 5    21 A
 6     9 B
 7    13 C
 8     5 D
 9     3 A
10     6 B
11     7 C
12    31 D
13     9 A
14    17 B
15     2 C
16    40 D
```

```
# Select all rows where AVAL > 10
filter(testdata,AVAL>10)

# Select all rows where AVAL > 10 and PARAMCD="C"
filter(testdata,AVAL>10,PARAMCD=="C")
```



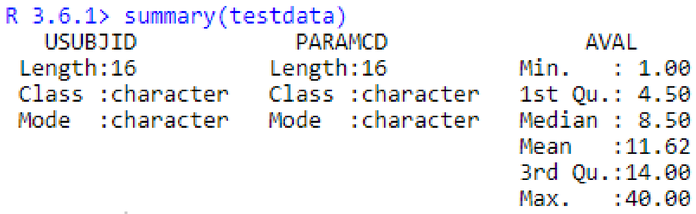
```
R 3.6.1> filter(testdata,AVAL>10,PARAMCD=="C")
# A tibble: 1 x 3
  USUBJID  PARAMCD  AVAL
  <chr>    <chr>   <dbl>
1 SUBJECT2 C          13
```

NOVARTIS | Reimagining Medicine

Data Review by R : Summarize the data

# Display the data summary

summary(testdata)

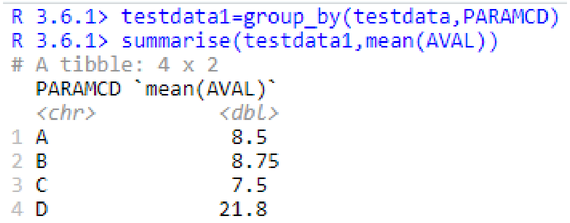


```
R 3.6.1> summary(testdata)
   USUBJID             PARAMCD               AVAL
 Length:16          Length:16          Min.   : 1.00
 Class :character   Class :character   1st Qu.: 4.50
 Mode  :character   Mode  :character   Median : 8.50
                                       Mean   :11.62
                                       3rd Qu.:14.00
                                       Max.   :40.00
```

# Calculate mean value by PARAMCD

testdata1=group_by(testdata,PARAMCD)

summarise(testdata1,mean(AVAL))



```
R 3.6.1> testdata1=group_by(testdata,PARAMCD)
R 3.6.1> summarise(testdata1,mean(AVAL))
# A tibble: 4 x 2
  PARAMCD `mean(AVAL)`
  <chr>          <dbl>
1 A               8.5
2 B               8.75
3 C               7.5
4 D              21.8
```

Function	Use
mean	Mean
min	Minimum
var	Variance
sum	Sum of values

Function	Use
median	Median
max	Maximum
sd	Standard Deviation
length	Length of values

Example of R programming for data creation

```
library(haven)
library(dplyr)

ex_1 <- read_sas("/***Path name ***/ex.sas7bdat", NULL)
suppex_1 <- read_sas(" /***Path name ***/suppex.sas7bdat", NULL)

#transpose
suppex_2 <- select(suppex_1, USUBJID, IDVARVAL, QNAM, QVAL)
suppex_3 <- spread(suppex_2, key=USUBJID+IDVARVAL, value=QVAL)

#Merge
ex_m1 <- merge(ex_1, suppex_3, by=c("USUBJID", "IDVARVAL"), all=T)

#AVAL, PARAMCD
ex_m2 <- select(ex_m1, USUBJID, EXSTDTC, EXDOSE)
ex_m3 <- group_by(ex_m2, USUBJID)
ex_m4 <- summarise(ex_m3,
                                              EXPWKS=(max(as.Date(substr(EXSTDTC, 1, 10))) - min(as.Date(substr(EXSTDTC, 1, 10))) + 28)/7,
                                              DOSTOTC=sum(EXDOSE), NUMDOSES=sum(EXDOSE > 0), NMISSDOSE=sum(EXDOSE == 0))
ex_m5 <- gather(ex_m4,key="PARAMCD", value="AVAL", EXPWKS, DOSTOTC, NUMDOSES, NMISSDOSE)

#PARAM
ex_m5$PARAM <- ifelse(ex_m5$PARAMCD == "EXPWKS", "Exposure (weeks)",
                                              ifelse(ex_m5$PARAMCD == "DOSTOTC", "Total cumulative dose (mg)",
                                              ifelse(ex_m5$PARAMCD == "NUMDOSES", "Total number of Doses",
                                              ifelse(ex_m5$PARAMCD == "NMISSDOSE", "Total number of missed Doses", NA))))

#AVALCAT1
ex_m5$AVALCAT1 <- ifelse(ex_m5$PARAMCD == "EXPWKS",
                                              {ifelse((ex_m5$AVAL > 0)&(ex_m5$AVAL <= 4), "> 0-4",
                                              ifelse((ex_m5$AVAL > 4)&(ex_m5$AVAL <= 8),"> 4-8",
                                                                        ⋮
                                              ifelse(ex_m5$AVAL > 48, "> 48",
                                              NA)))))))))))))},NA)

#sorting
ex_m6 <- ex_m5[c(1,7,4,2,8,3,5,6)]
```

NOVARTIS | Reimagining Medicine

Example of R programming for data creation

```r
library(haven)
library(dplyr)
```

Import the Packages; **haven** (Import and Export 'SPSS', 'Stata' and 'SAS' Files) and **dplyr** (To handle the data frame)

```r
ex_1 <- read_sas("/***Path name ***/ex.sas7bdat", NULL)
suppex_1 <- read_sas(" /***Path name ***/suppex.sas7bdat", NULL)
```

Import SAS Files.

```r
#transpose
suppex_2 <- select(suppex_1, USUBJID, IDVARVAL, QNAM, QVAL)
suppex_3 <- spread(suppex_2, key=USUBJID+IDVARVAL, value=QVAL)
```

Spread a key-value pair across multiple columns.

```r
#Merge
ex_m1 <- merge(ex_1, suppex_3, by=c("USUBJID", "IDVARVAL"), all=T)
```

Merge Two Data Frames.

```r
#AVAL, PARAMCD
ex_m2 <- select(ex_m1, USUBJID, EXSTDTC, EXDOSE)
ex_m3 <- group_by(ex_m2, USUBJID)
ex_m4 <- summarise(ex_m3,
                                        EXPWKS=(max(as.Date(substr(EXSTDTC, 1, 10))) - min(as.Date(substr(EXSTDTC, 1, 10))) + 28)/7,
                                        DOSTOTC=sum(EXDOSE), NUMDOSES=sum(EXDOSE > 0), NMISSDOSE=sum(EXDOSE == 0))
ex_m5 <- gather(ex_m4,key="PARAMCD", value="AVAL", EXPWKS, DOSTOTC, NUMDOSES, NMISSDOSE)
```

Create the parameters: the total dose, the number of doses, and the number of missed doses

Gather columns into key-value pairs.

```r
#PARAM
ex_m5$PARAM <- ifelse(ex_m5$PARAMCD == "EXPWKS", "Exposure (weeks)",
                                        ifelse(ex_m5$PARAMCD == "DOSTOTC", "Total cumulative dose (mg)",
                                        ifelse(ex_m5$PARAMCD == "NUMDOSES", "Total number of Doses",
                                        ifelse(ex_m5$PARAMCD == "NMISSDOSE", "Total number of missed Doses", NA))))
```

Define PARAM corresponding to PARAMCD

```r
#AVALCAT1
ex_m5$AVALCAT1 <- ifelse(ex_m5$PARAMCD == "EXPWKS",
                                        {ifelse((ex_m5$AVAL > 0)&(ex_m5$AVAL <= 4), "> 0-4",
                                        ifelse((ex_m5$AVAL > 4)&(ex_m5$AVAL <= 8),"> 4-8",
                                                            ⋮
                                        ifelse(ex_m5$AVAL > 48, "> 48",
                                        NA)))))))))))))},NA)
```

Define AVALCAT1 corresponding to AVAL

```r
#sorting
ex_m6 <- ex_m5[c(1,7,4,2,8,3,5,6)]
```

Ordering the variables

NOVARTIS | Reimagining Medicine

Key differences between SAS and R

Import SAS File

```
library(haven)
Dataset Name <- read_sas("/***Path name ***/SAS File Name.sas7bdat")
```

Package **haven** import external statistical methods into R via the embedded 'ReadStat' C library. **read_sas()** Read SAS files into R.

Spread and Gather

```
spread(Dataset Name, key=***, value=***)
gather(Dataset Name, key=***, value=***)
```

Spread function Spread a key-value pair across multiple columns. **Gather function** Gather columns into key-value pairs.

	COl
AA	1
AA	2
AA	3
BB	4
BB	5
BB	6

	Col 1	Col 2	Col 3
AA	1	2	3
BB	4	5	6

	Col 1	Col 2	Col 3
AA	1	2	3
BB	4	5	6

	COl
AA	1
AA	2
AA	3
BB	4
BB	5
BB	6

```
///SAS///
proc transpose data=Dataset Name out=Dataset Name prefix=***;
                    var ***;
                    by ***;
run;
```

Key differences between SAS and R

Merge

Dataset Name*<- merge(*Dataset A, Dataset B,* by=)**

Merge function merge two datasets horizontally. Sort is not needed.

id	COl1
AA	1
BB	2
CC	3
DD	4
EE	5

+

id	COl2
AA	1999
FF	2000
DD	2001
CC	2002
EE	2003

by "id" →

id	COl1	Col2
AA	1	1999
BB	2	2004
CC	3	200

```
///SAS///
proc sort data=Dataset A; by ***;run; /*sort logic is needed*/
proc sort data=Dataset B; by ***;run;
Data Dataset Name;
                    merge dataset A dataset B;
                    by ***;
run;
```

Ordering the variable

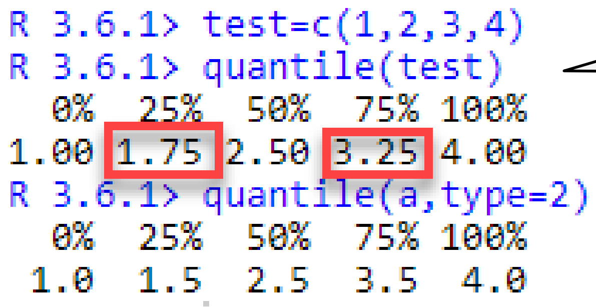
***Dataset Name* <- *Dataset Name*[c(*new order 1, new order 2,* …)]**

In R, reordering a column is very simple.

```
///SAS///
Data Dataset Name;
                    length (or format) value 1 value 2 …;
run;
```

Key differences between SAS and R

Function parameters - quantile

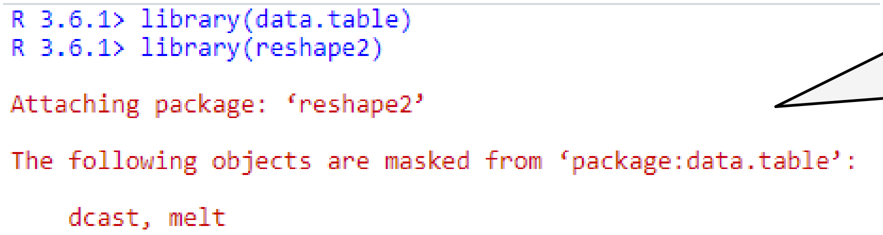


```
R 3.6.1> test=c(1,2,3,4)
R 3.6.1> quantile(test)
  0%  25%  50%  75% 100% 
1.00 1.75 2.50 3.25 4.00 
R 3.6.1> quantile(a,type=2)
  0%  25%  50%  75% 100% 
 1.0  1.5  2.5  3.5  4.0 
```

Type=7 (Default)

Type=2 (SAS equivalent)

Function masking



```
R 3.6.1> library(data.table)
R 3.6.1> library(reshape2)

Attaching package: ‘reshape2’

The following objects are masked from ‘package:data.table’:

    dcast, melt
```

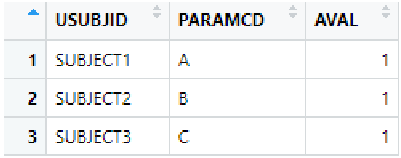
Both library have dcast and melt function. Recommend to use: data.table::dcast

NOVARTIS | Reimagining Medicine

Compare logic by R programming

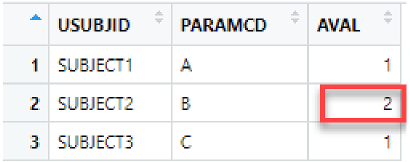
```
summary(arsenal::comparedf(x = dev, y = qc, by = c(“USUBJID",“PARAMCD")))
```

dev



	USUBJID	PARAMCD	AVAL
1	SUBJECT1	A	1
2	SUBJECT2	B	1
3	SUBJECT3	C	1

qc



	USUBJID	PARAMCD	AVAL
1	SUBJECT1	A	1
2	SUBJECT2	B	2
3	SUBJECT3	C	1

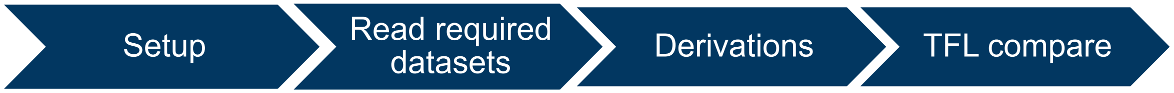
```
statistic                                                    value
------------------------------------------------------------  ------
Number of by-variables                                            2
Number of non-by variables in common                              1
Number of variables compared                                      1
Number of variables in x but not y                                0
Number of variables in y but not x                                0
Number of variables compared with some values unequal             1
Number of variables compared with all values equal                0
Number of observations in common                                  3
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       1
Number of observations with all compared variables equal          2
Number of values unequal                                          1

Table: Differences detected

var.x    var.y    USUBJID     PARAMCD    values.x    values.y     row.x    row.y
------   ------   ---------   --------   ---------   ---------   ------   ------
AVAL     AVAL     SUBJECT2    B          1           2                2        2
```

NOVARTIS | Reimagining Medicine

Key information of ADaM validation by R

- R is useful for data review
- Difference between SAS and R
  - R programming is simple code
  - Different default statistical method
  - There are some R specific logics compared to SAS
    - Library
    - spread and gather
    - Ordering the variables
- Compare logic can be used similar to SAS

Overview of TFLs Validation



Setup program

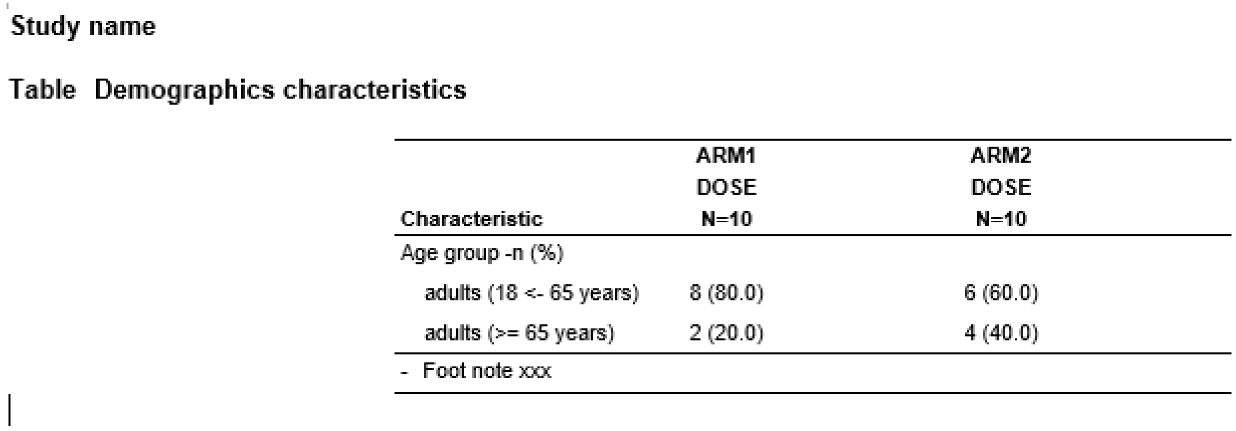
- All TFL QC programs use Setup program
- Clearing environment
- Unload all packages
- Setting paths to study folders
- Load some useful libralies
  - pacman: efficient loading and unloading of packages
  - haven: importing SAS datasets
  - dplyr, tidyr, data.table: more efficient handling of datasets
  - sqldf: functions to use SQL syntax in R
  - stringr, lubridate: functions for data types handling
  - striptf, arsenal: import rtf files and compare datasets

NOVARTIS | Reimagining Medicine

TFLs QC compare process: Pattern1

- In case of word cell based format TFL

Example of Table



Study name

Table Demographics characteristics

Characteristic	ARM1 DOSE N=10	ARM2 DOSE N=10
Age group -n (%)		
adults (18 <- 65 years)	8 (80.0)	6 (60.0)
adults (>= 65 years)	2 (20.0)	4 (40.0)

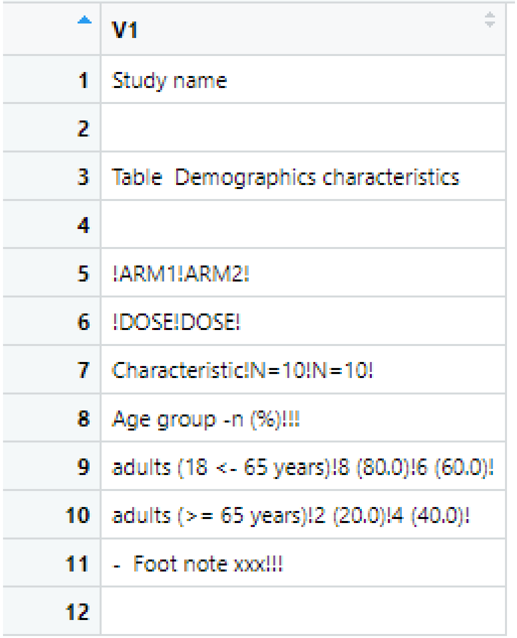
- Foot note xxx

NOVARTIS | Reimagining Medicine

TFLs QC compare process: Pattern1

STEP 1: Read the rtf file and insert '!' at the end of cell

```
rtfds1 = data.table(read_rtf('reports/saf/test.rtf',
row_start = "", cell_end = "!"))
```



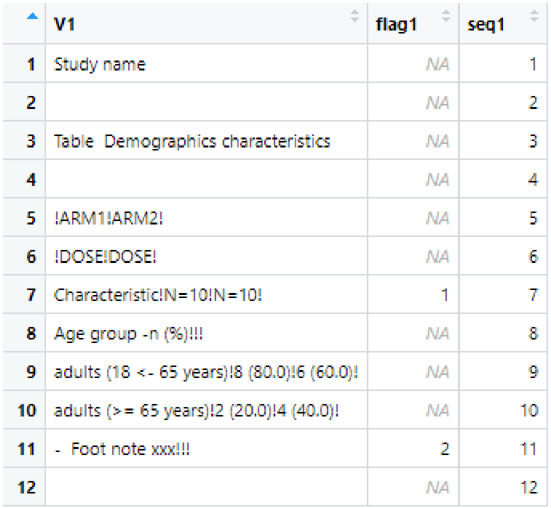
	V1
1	Study name
2	
3	Table Demographics characteristics
4	
5	!ARM1!ARM2!
6	!DOSE!DOSE!
7	Characteristic!N=10!N=10!
8	Age group -n (%)!!!
9	adults (18 <- 65 years)!8 (80.0)!6 (60.0)!
10	adults (>= 65 years)!2 (20.0)!4 (40.0)!
11	- Foot note xxx!!!
12	

NOVARTIS | Reimagining Medicine

TFLs QC compare process: Pattern1

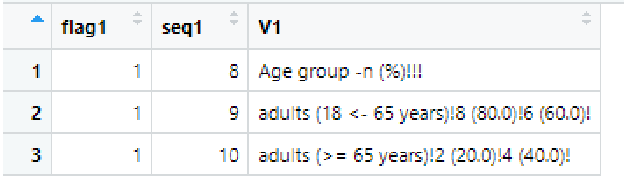
STEP 2: Remove non-data rows

```
rtfds1[, ':='(flag1 = case_when(grepl('Characteristic', V1, fixed = T) ~ 1, grepl('Foot note',
V1, fixed = T) ~ 2), seq1 = .I)]
rtfds2 = rtfds1[is.na(flag1) == 0, .(flag1, seq1)][copy(rtfds1)[,flag1 := NULL], on = .(seq1),
roll = T][flag1 == 1 & grepl('Characteristic', V1, fixed = T) == 0]
```



	V1	flag1	seq1
1	Study name	NA	1
2		NA	2
3	Table Demographics characteristics	NA	3
4		NA	4
5	!ARM1!ARM2!	NA	5
6	!DOSE!DOSE!	NA	6
7	Characteristic!N=10!N=10!	1	7
8	Age group -n (%)!!!	NA	8
9	adults (18 <- 65 years)!8 (80.0)!6 (60.0)!	NA	9
10	adults (>= 65 years)!2 (20.0)!4 (40.0)!	NA	10
11	- Foot note xxx!!!	2	11
12		NA	12

Target of QC



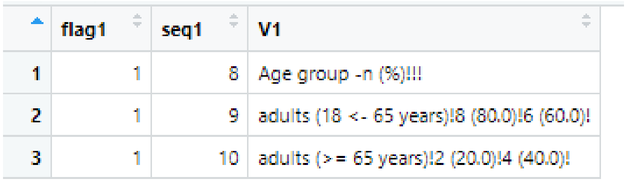
	flag1	seq1	V1
1	1	8	Age group -n (%)!!!
2	1	9	adults (18 <- 65 years)!8 (80.0)!6 (60.0)!
3	1	10	adults (>= 65 years)!2 (20.0)!4 (40.0)!

NOVARTIS | Reimagining Medicine

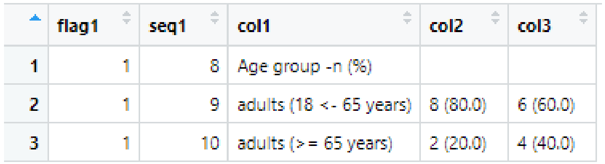
TFLs QC compare process: Pattern1

STEP 3: Separate data into separate columns

```
rtfds3 = dplyr::mutate(rtfds2,V1=substr(V1,1,nchar(V1)-1))
rtfds4 = separate(rtfds3, V1, paste('col',1:(unique(stri_count(rtfds2$V1,fixed = '!')) + 0), sep =
''), sep = "!")
```



	flag1	seq1	V1
1	1	8	Age group -n (%)!!!
2	1	9	adults (18 <- 65 years)!8 (80.0)!6 (60.0)!
3	1	10	adults (>= 65 years)!2 (20.0)!4 (40.0)!



	flag1	seq1	col1	col2	col3
1	1	8	Age group -n (%)		
2	1	9	adults (18 <- 65 years)	8 (80.0)	6 (60.0)
3	1	10	adults (>= 65 years)	2 (20.0)	4 (40.0)

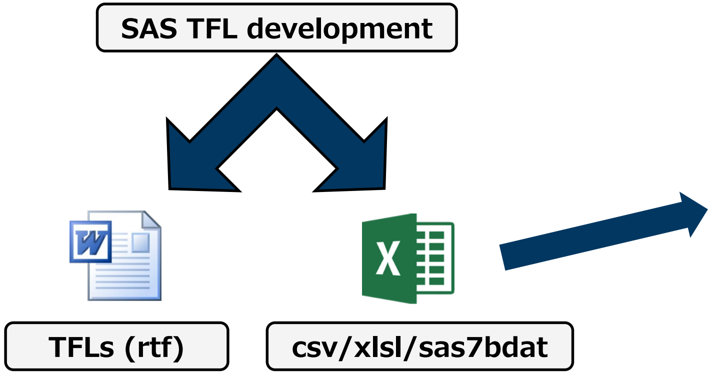
STEP 4: Compare RTF data and QC data

NOVARTIS | Reimagining Medicine

TFLs QC compare process: Pattern2

- Other than word cell based format TFL
- SAS TFL development program generate the csv/xlsx/sas7bdat as well
- R QC program read the csv/xlsx/sas7bdat and compare it

Example of the process



R QC program

- Develop the QC result data same structure of SAS TFL
- Read csv/xlsl/sas7bdat
- Compare the data

NOVARTIS | Reimagining Medicine

Key information of TFL validation by R

- Pattern1: Word cell based format
  - ➢ rtf file can be directory read by R
- Pattern2: Other than word cell based format
  - ➢ Need to export the data from developer
- Compare logic can be used for both pattern

Key takeaways

- There are some different logics between SAS and R
- R is useful for data review
- Compare logic is useful for ADaM and TFL validation

Thank you

NOVARTIS | Reimagining Medicine