**Paper CT10**

# Mr. (R)ight – Why R Should Be Used for Validating TLFs in Clinical Trials?

Monika Ludwinska, Intego Group, Warsaw, Poland

## ABSTRACT

SAS is widely used in validating TLFs in clinical trials but over the past few years R has emerged as an alternative software to produce high quality outputs. This open source language created for statistical computing is gaining importance in pharmaceutical industry because of its flexibility and variety. Quality control process is needed to ensure the accuracy of outcomes by comparing results prepared by two independent programmers based on given specification.

This paper will look at R and its strengths in the validation process of tables, listings and figures. It will focus on advantages of using another language in the quality control process and will practically show how validation is performed. Moreover, it will present important R functions which every programmer should be familiar with in daily work to efficiently lead the quality process. Furthermore, pitfalls will be exposed while validating SAS outputs with R software on examples.

## INTRODUCTION

In the evolving world, programmers have to be prepared that their knowledge should be constantly ready to be broaden. Recent growth of R usage over the past couple of years is not something brand new for pharmaceutical industry workers. Every newcomer in this business and in the programming field has been acquainted with SAS ®. Starting with this programming language as the main engine for creation and validation of outputs for submissions' needs is what is expected at the beginning. Usage of R in the validation part of the process flow is a wide-ranging debate including a number of benefits but primarily about utilization of the open source language to operate with human data. The industry develops rapidly and if a programmer wants to be prepared for the market demands, R knowledge is essential to meet business' criteria. The additional reason that should be considered is self-esteem and confidence. Being educated in a new programming language is an asset for current and future employers and it generates an increase in motivation and seeking new challenges.

SAS programmers are still highly required, though we can see a growing number of employment opportunities where R language is an advantage or even a must. We have to be conscious of this evolution of the market. Besides market requirements, programmers also need to reflect on their own motivation - should I only be involved with SAS? Is R something that I really need? Based on experience and current knowledge, software engineers are supposed to have a clear career path and designated steps to make.

Speaking about the Quality Control process, firstly the difference between QC (Quality Control) and validation should be discussed. Validation is part of a software development process and validation of computerized systems, while QC is checking the data and is part of validation. Validation is based on Food and Drug Administration guidance document "Part 11: Electronic Records, Electronic Signatures". This document emphasizes the need for validation and work qualifications to satisfy the regulatory bodies and fundamental elements of quality for public safety. Good Clinical Practice (GCP) is another document which discusses the need of validation in clinical trials. It reviews overall standards for implementing a clinical trial. It also ensures that the generated data is prepared in a controlled manner and has integrity.

Quality Control (QC) in clinical trials should be conducted on each step of the study. It continues after the trial has been finished and has a significant role to ensure that produced Tables, Listings and Figures (TLF) are the highest quality possible. QC of deliverables ensures the protection of human subjects in clinical trials. This is a fundamental process of guaranteeing that the generated output is accurate by comparing it with an independently prepared program by a second line programmer. Quality Control procedure should include not only reviewing the output, but also verifying the data used for creation. This process provides finding discrepancies between two autonomous programs and programmers. It proves that the outcome accurately and effectively represents the original source of clinical study data. This is a justification of used methods and an engine to achieve the outcome. The Quality Control process requires more effort and accuracy from a programmer.

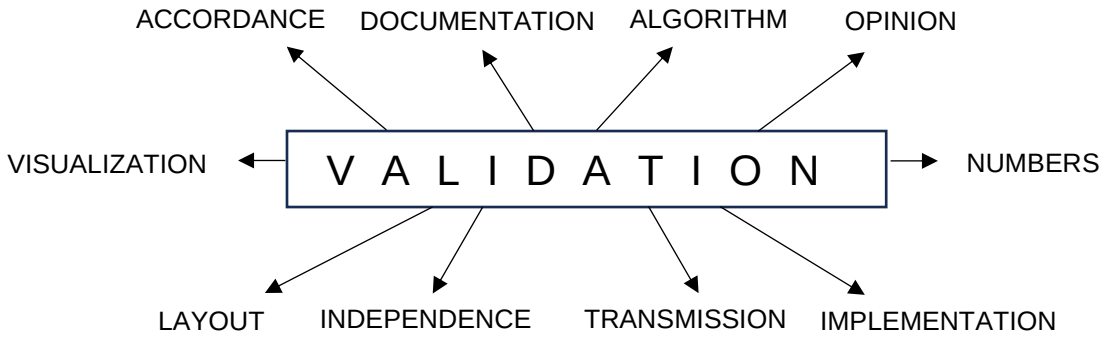
This paper is not focused on convincing programmers and managers to use R in QC procedures, taking into account popularity of this open source language. The aim of this work is to show principles of quality control and how to accomplish it in R. **Additionally, in this work the word "validation" will be used to refer to Quality Control procedure.**

## PRINCIPLES OF INDEPENDENT VALIDATION FOR QUALITY CONTROL PROCEDURE

Programmers and their work plays a crucial role in clinical trials. Team members who deal with a specific study are aware that the data is human-based and have to work with it scrupulously taking into consideration that it may influence future treatment and lives of people across the world. Their goal is to support development of new medicines so quality control is a significant procedure for programmers.

This work is focused on TLFs, so the mentioned principles are not the only ones which should be followed for ADaM QC purpose. This paper is focused on usage of R in QC procedure but the principles below are valid for any programming language used by second line programmers.

To picture better the quality control procedure, a list of principles was created and described below in the model of **Validation**.




### VISUALIZATION

Besides the used data and justifying their usage, visual aspect of the output needs to be analyzed. The outcome has to be thoroughly checked. Basically, every output is created based on the provided shell which should be followed. Results need to be consistent with the Statistical Analysis Plan and other specifications. As specified by the provided documentation, the programmer has to check the following: the titles, footnotes, order of treatment groups and categories, used symbols, population, headers, units, precision, page breaks, column spacing, alignment, labels, categories, special characters and other important things. Additionally for figures, colors and axes have to be verified. All possible values should be covered on figure along with outliers and extreme values. The list of things to be checked is long and the programmer must be ready to cover all the checks before marking the output as passed.

### ACCORDANCE

"Accordance" in quality control procedure means keeping consistency across all TLFs. Two parts of consistency can be distinguished:

- Consistency between related outputs (Table vs Figure etc.)
- Consistency between type of output (All Figures or all Tables or all Listings)

Taking the first point as an example, there is a high possibility that for a newly created figure, there is a corresponding table (and vice versa) and values across both related outputs have to be the same and should be displaying the same values. For the second type of accordance, for instance, the colors checks should be performed to keep colors for treatment/laboratory/race/country (and so on) categories the same across all the figures. Additionally, the symbols used and type of lines have to be unified along with values in a legend. For the tables and listings programmers should be aware of keeping unification between the alignment, displayed units, precision etc.

### LAYOUT

Starting preparation of the validation program should be preceded by preparing a layout of validation. It is crucial to have a specified plan before initiation of programming. Gathering all the information from documents is crucial to prepare a clear code from the starting point. The provided deadline has to be also checked if it is feasible to achieve. All possible obstacle should be reported to Project Leader.

### INDEPENDENCE

To ensure the best quality of QC process, every second line programmer should get ready with an individually prepared program to justify the results generated by the first line programmer with usage of the same inputs and requirements. An autonomous QC should not be focused on achieving clean compare results, it has to rather be pointed at comparing differences and hence errors in deliverables. Every programmer's logic is different and various methods they use should lead to getting the same results. The second line programmer has to capture any odds in the data that could be missed by the first line programmer. These programs need to include defensive checks to find issues in the source data. Once differences have been found and identified, the decision has to be made – which side is correct? The first or the second line? Copying and looking into the first line programmer's program is forbidden.

#### DOCUMENTATION

Quality Control procedure should be well documented with logs (from both sides), clean codes and the compare file which guarantees equivalence. The logs have to be clear from errors and also from warnings unless there is a valid reason for it, for example standard macros in the Company. Also some SAS notes have to be avoided. Every code should be easy to read and understand. It should contain enough comments to easily understand every step of it. Hard coding is not generally allowed, it only could take place in specific and confirmed situations.

#### ALGORITHM

Creating a program in the quality control procedure requires usage of correct algorithms to get proper values. The programmer has to be sure about the chosen logic and data manipulations if it is appropriate and consistent with the Statistical Analysis Plan or other applicable specifications. Correct formulas and statistic functions have to be used along with proper variables, flags, population, selected baseline etc. Usage of a proper statistical model has to be justified and verified with specification.

#### TRANSMISSION

Transmission in the model of validation means communication with the first line programmer. Programmers frequently work in different time zones, so keeping each other informed about the results and differences should take place on a regular basis. First line programmers should be informed mainly of the differences explained on the data supported by the documentation and with details on which programming side is correct. As programmers do not work only on one specific output at one time, comments from second line programmers should be brief, but supported by the data from ADaM. Additionally, communication with Project Leaders has to be provided with information about potential data issues.

#### IMPLEMENTATION

Industry requirements and standards should be applied to a specific output. The validator has to critically evaluate the produced output against the rules and principles. During creation of the program, programmers need to review the Statistical Analysis Plan, Protocol, shell and the Company's standards. All the factors should match between the documents and have to be implemented in the code. Programmers also need to verify if all the required variables are calculated within ADaM datasets. TLF programmers have to avoid calculating their own variables in their codes, for instance, new categories. Each TLF should be able to be produced from the "analysis-ready" ADaM dataset.

#### OPINION

The idea behind the word "opinion" is providing a critical opinion not only for production programmers, but also for ourselves, validators. If mismatches are encountered, firstly, a validator has to be critical about their outcomes. Every result needs be justified in the data and self-validation should be performed before analyzing the outcome and discrepancies.

#### NUMBERS

This part of the validation model means proper displaying of values. Missing values should be precisely checked along with categorical and continuous data. Statistics have to be thoroughly verified and examined against missing values and procedures. Numeric data is reported with precision necessary for clinical interpretation, that is why displays have to reflect what is stored in the datasets, for instance, programmers have to avoid the truncation: "1.23" should not be cut to "1". Additionally, the following checks have to be verified: decimals, denominators, leading zeros, scientific notation, precision of percentages, p-values, confidence intervals etc.

## R AS A VALIDATION TOOL

R consists of a large number of packages and more and more pharmaceutical companies have started to create their own for validation and producing outputs. To start working in R as a QC tool, several packages should be loaded into an R session.

1. haven – loading SAS data formats (sas7bdat)
2. dplyr, tidyr – data manipulations
3. diffdf – comparison package
4. logrx – generating log file

Haven, dplyr and tidyr packages are part of tidyverse - one of the most popular and opinionated collection of packages frequently used by pharmaceutical companies. Logrx facilitates creating log files and diffdf is a package used for comparing results – both are supported by pharmaverse so another set of packages for the pharma-specific environment.

## TABLES

Tables are the most commonly produced deliverable for programmers. Summary tables display statistics which can be interpreted for study purposes. The minimum set of summary statistics should be included: n, mean, median, standard deviation, minimum and maximum to give a general description of the distribution.

Below is the Baseline Characteristics table prepared in SAS. It includes only BMI statistics and age group category counts with percentages. It shows treatments along with the total group prepared for the enrolled population. The table contains a row for group which is missing in the data (>=85). This age group is in the documentation and has to be provided on the display with '0' for all treatment groups.

Population: Enrolled

Table 1
Summary of Baseline Characteristics

| Characteristic | | Treatment X (N=9) | Placebo (N=8) | Total (N=17) |
|---|---|---|---|---|
| BMI (kg/m^2) at Baseline | n | 9 | 8 | 17 |
| | Mean | 24.534 | 26.461 | 25.441 |
| | SD | 5.1625 | 5.6305 | 5.3083 |
| | Median | 24.570 | 27.020 | 25.070 |
| | Min. | 16.67 | 19.11 | 16.67 |
| | Max. | 35.51 | 35.38 | 35.51 |
| Age group (%) | n | 8 | 7 | 15 |
| | <18 | 2 (25%) | 1 (14%) | 3 (20%) |
| | >=18-64 | 3 (38%) | 4 (57%) | 7 (47%) |
| | >=65-84 | 3 (38%) | 2 (29%) | 5 (33%) |
| | >=85 | 0 | 0 | 0 |

Note: This is footnote 1.
Note: This is footnote 2.

Firstly, ADSL dataset and the production dataset of table have to be loaded into R with the haven package. In the next step, the total group has to be added, which can be easily done with *mutate* and *bind_rows* functions. Afterwards, the total number of subjects in each treatment group can be calculated (**bigN** variable) with *group_by* and *mutate*. Additionally, **label** variable is calculated with *paste0* with pattern: "YYY~(N=XX)". This variable will be further used for dynamic assignment of labels to variables.

```
tab <- adsl %>%
  mutate(TRT01P = "Total",
         TRT01PN = 9) %>%
  bind_rows(adsl) %>%
  filter(ENRLFL == "Y") %>%
  #BigN
  group_by(TRT01P) %>%
  mutate(bign = n(),
         label = paste0(TRT01P, "~(N=", bign, ")"))
```

The first part of the table consists of statistics for continuous variable BMIBL. Statistics can be calculated with *summarise* (*summarize*) function and respectively *n, mean, sd, median, min* and *max* functions. Values from SAS dataset come from character variables so *as.character* function is used to change numeric to character. Minimum and maximum variables are stored with the same number of decimal places as a collected precision, mean and median with one additional decimal place and standard deviation with two additional decimal places. *Round* function removes trailing zeros so, for example, value 24.57001 rounded to three decimal places will display 24.57, not 24.570 so *format* function has to be used to add trailing zero.

```
qc1 <- tab %>%
  group_by(TRT01PN) %>%
  summarise(n = as.character(n()),
            Mean = as.character(format(round(mean(BMIBL), 3), nsmall = 3)),
            SD = as.character(format(round(sd(BMIBL), 4), nsmall = 4)),
            Median = as.character(format(round(median(BMIBL), 3), nsmall = 3)),
            Min. = as.character(min(BMIBL)),
            Max. = as.character(max(BMIBL))) %>%
  pivot_longer(cols = -TRT01PN, names_to = "stats") %>%
```

```
  pivot_wider(values_from = value, names_from = TRT01PN, names_prefix = "col") %>%
  mutate(char = "BMI (kg/m^2) at Baseline")
```

Manipulating data consists of two pivot functions Firstly, *pivot_longer* ia used to move statistics to one column and then *pivot_wider* to move treatment groups into separated columns.

| TRT01PN | n | Mean | SD | Median | Min. | Max. |
|---|---|---|---|---|---|---|
| 1 | 9 | 24.534 | 5.1625 | 24.570 | 16.67 | 35.51 |
| 2 | 8 | 26.461 | 5.6305 | 27.020 | 19.11 | 35.38 |
| 9 | 17 | 25.441 | 5.3083 | 25.070 | 16.67 | 35.51 |

initial dataset

| TRT01PN | stats | value |
|---|---|---|
| 1 | n | 9 |
| 1 | Mean | 24.534 |
| 1 | SD | 5.1625 |
| 1 | Median | 24.570 |
| 1 | Min. | 16.67 |
| 1 | Max. | 35.51 |
| … | … | … |

pivot_longer result

| stats | col1 | col2 | col9 |
|---|---|---|---|
| n | 9 | 8 | 17 |
| Mean | 24.534 | 26.461 | 25.441 |
| SD | 5.1625 | 5.6305 | 5.3083 |
| Median | 24.570 | 27.020 | 25.070 |
| Min. | 16.67 | 19.11 | 16.67 |
| Max. | 35.51 | 35.38 | 35.51 |

pivot_wider result

The second part of the table is split into two data frames. The first one easily calculates n values and with usage of *pivot_wider* gets values into columns.

```
qc2_1 <- tab %>%
  filter(!is.na(AAGEGR1N)) %>%
  group_by(TRT01PN) %>%
  summarise(n=as.character(n())) %>%
  pivot_wider(values_from = n, names_from = TRT01PN, names_prefix = "col") %>%
  mutate(char = "Age group (%)",
         stats = 'n')
```

qc2_2 data frame is tricky because of the missing age group in ADSL Dummy dataset has to be produced and it needs to contain combinations of all age groups. This dummy dataset can be merged so an additional row with group “>=85” will be included. For percentages, **smallN** is used as a denominator which is calculated in a separated data frame and also merged with qc2_2.

```
#Denominator
smallN <- tab %>%
  filter(!is.na(AAGEGR1N)) %>%
  group_by(TRT01PN) %>%
  summarise(smallN=n())

#Dummy dataset
dummy_aagegr1 <- data.frame(AAGEGR1 = c("<18", ">=18-64", ">=65-84",">=85"),
                            AAGEGR1N = c(1, 2, 3, 4))

qc2_2 <-  merge(tab, smallN, by = "TRT01PN") %>%
  filter(!is.na(AAGEGR1N)) %>%
  group_by(TRT01PN, AAGEGR1, AAGEGR1N, smallN) %>%
  summarise(n = n()) %>%
```

```
  mutate(value = as.character(paste0(as.character(n), " (",
as.character(round(n/smallN*100)), "%)"))) %>%
  pivot_wider(id_cols = c(AAGEGR1, AAGEGR1N), values_from = value, names_from =
TRT01PN, names_prefix = "col") %>%
  #Merge with dummy
  merge(dummy_aagegr1, all.y = TRUE) %>%
  rename(stats = AAGEGR1) %>%
  mutate(char = "Age group (%)") %>%
  select(-AAGEGR1N)
```

After merging the dummy, *NA* values occur in the dataset so it has to be replaced with “0”.

```
qc2_2[is.na(qc2_2)] <- "0"
```

Prepared data frames: qc1, qc2_1 and qc2_2 can be gathered together with *bind_rows* function. The final step of QC dataset is setting labels dynamically. The variable **label** was prepared at the very beginning of programming. It can be now pulled from the data into vector with function *pull*. Treatment numeric variable is selected from the data to sort it and pull labels into vector in the proper order. Further, labels are assigned to attribute “label”.

```
labels <- tab %>%
  distinct(label, TRT01PN) %>%
  arrange(TRT01PN) %>%
  pull(label)

attr(qc$col1, "label") <- labels[1]
attr(qc$col2, "label") <- labels[2]
attr(qc$col9, "label") <- labels[3]
```

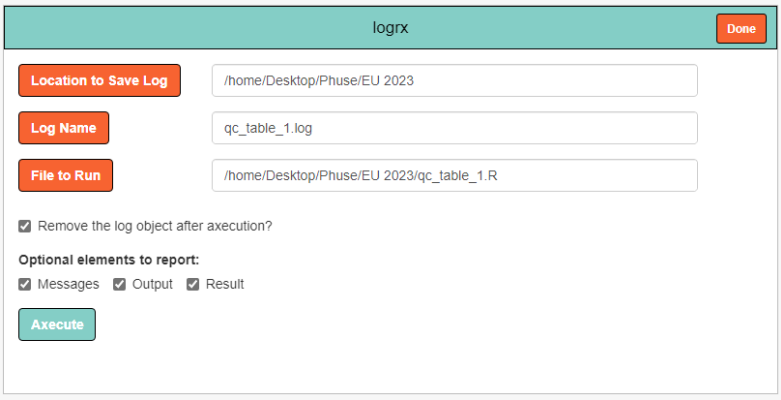
Production dataset may include spaces which are used for proper alignment of the table. Firstly, the second line programmer has to examine if the table is produced with proper numbers, so to make this task easier, all spaces can be removed from the production and QC datasets with *lapply* and *gsub* functions. Appropriate alignment of values in the table has to be manually verified as the second step of the table’s validation.

```
prod <- as.data.frame(lapply(prod, function(y) gsub("\\s+", "", y)))
qc <- as.data.frame(lapply(qc, function(y) gsub("\\s+", "", y)))
```

*Diffdf* function can be used to produce lst file with comparison results.

```
diffdf(base = prod,
       compare = qc,
       file = "qc_table_1.lst")
```

To produce a log file from an R program, after installing logrx package and importing it into R session, it can be obtained from Addins toolbar menu in R Studio or can be run directly with *axecute* function, for instance *axecute("table_1.R")*.

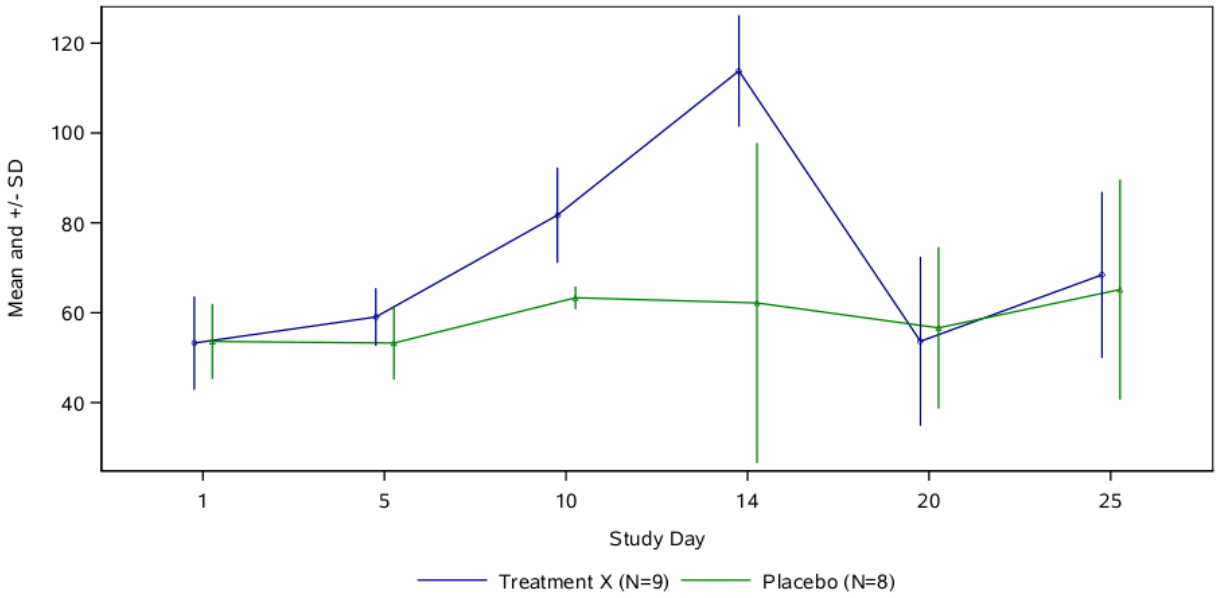


## FIGURES

So far there has not been a specific R package for figure validation, especially to validate the visual aspect of the figure, for instance, if proper colors were used for a specific group, appropriate marker symbols, checking axes ranges. QC of figures should consist of, firstly dataset validation, which is directly used to create a figure by the first line programmer. This dataset has to be available for QC purpose and should be verified in the same manner as a table or a listing. An example of a laboratory figure is provided below along with the dataset prepared in SAS.

Population: Safety

Figure 1
Mean (SD) Creatinine Values Over Time by Treatment Group




Firstly, the same as for tables, number of patients in treatment groups has to be calculated (**bignN**) based on ADSL dataset for safety population. **Trtlabel** is calculated with **bign** and is included in the legend of figure.

```
bign <- adsl %>%
  filter(SAFFL == "Y") %>%
  group_by(TRT01A) %>%
  summarise(bign = n()) %>%
  mutate(trtlabel = paste0(TRT01A, " (N=", bign, ")"))
```

This data frame is further merged with the laboratory ADaM dataset. QC data frame is grouped by several variables and statistics are calculated.

```
fig <- merge(adlb, bign, by = "TRT01A", all.x = TRUE) %>%
  filter(SAFFL == "Y")

qc <- fig %>%
  group_by(TRT01AN, TRT01A, trtlabel, PARAM, ADY, bign) %>%
  summarise(mean = mean(AVAL),
            sd = sd(AVAL)) %>%
  mutate(uppersd = ifelse(!is.na(mean) & !is.na(sd), mean + sd, NA),
         lowersd =ifelse(!is.na(mean) & !is.na(sd), mean - sd, NA)) %>%
  mutate(mean = round(mean,8),
         uppersd = round(uppersd,8),
         lowersd = round(lowersd,8)) %>%
  select(-sd, -bign)
```

Production dataset which comes from SAS for figures does not have variables for axes as characters so it's not necessary to remove spaces before performing *diffdf* function.

```
diffdf(base = prod,
       compare = qc,
       file = "qc_figure_1.lst")
```

When the dataset for figure creation is validated, the second line programmer can start validating details of the figure. Firstly, x and y axes ranges can be checked per page. To verify this, the dataset can be grouped by a parameter and treatment and min and max from x/y values can be examined. Checking axes ranges will verify if all the data points are displayed in the figure.

```
 y <- qc %>%
  group_by(PARAM, TRT01A) %>%
  summarise(ymin = min(lowersd),
            ymax = max(uppersd))
```

| PARAM | TRT01A | ymin | ymax |
|---|---|---|---|
| Creatinine | Placebo | 26.53478 | 97.80922 |
| Creatinine | Treatment X | 34.81289 | 126.23201 |

R has a package imager which allows to load PNG images into R with function *load.image*. RGB channels are extracted from the image into the data frame. Further, *rgb* function is used to create colors in hexadecimal format. Later, the data frame is grouped, summarized and ordered to get most frequently used colors in the figure. As a result, white and black color occur at the top but in the third and fourth position there are colors used for the treatment groups. The obtained colors can be now checked against the produced figure. There is also a possibility that for figure creation an attribute dataset was used in SAS and is stored in the project's area so colors can be verified with this dataset.

```
library(imager)

colors_ = load.image("figure_1.png")

colors1 <- data.frame(
  R = as.vector(colors_[,,1]),
  G = as.vector(colors_[,,2]),
  B = as.vector(colors_[,,3]))

colors <- data.frame(RGB = rgb(colors1)) %>%
  group_by(RGB) %>%
  summarise(n = n()) %>%
  arrange(desc(n)) %>%
  head(4)
```

| RGB | n | color |
|---|---|---|
| #FFFFFF | 564684 | |
| #000000 | 4842 | |
| **#008000** | **775** | |
| **#00008B** | **579** | |

**LISTINGS**

Listings are the easiest deliverable to be produced in a study where almost no data manipulations are required. Listings should display the data as it is but in proper order. The example below provides Listing of Demographic Characteristics, which is an output prepared from ADSL dataset. Data manipulations only require filtering for proper population, calculation of the group variable and sorting – in the example below, the output is prepared for any site id and is sorted by this variable and the treatment variable.

```
qc <- adsl %>%
  filter(SAFFL == "Y") %>%
  mutate(group = paste0("Site Id.: ", SITEID)) %>%
  select(TRT01P,USUBJID,BRTHDTC,AAGE,SEX,HEIGHTBL,WEIGHTBL,group,TRT01PN,SITEID) %>%
  arrange(SITEID, TRT01PN)
```

Population: Enrolled

Listing 1
Listing of Demographic Characteristics

Site Id.: XYZ

| Treatment | Unique Subject Identifier | Date of Birth | Age (YEARS) [1] | Sex | Height (cm) | Weight (kg) |
|---|---|---|---|---|---|---|
| Treatment X | 001 | 1991 | 30 | F | 169 | 54.5 |
| | 002 | 1979 | 41 | F | 175 | 89.2 |
| | 004 | 1981 | 40 | F | 145 | 39.2 |
| | 009 | 1973 | 48 | M | 185 | 102.5 |
| | 010 | 1977 | 43 | M | 165 | 98.9 |
| | 011 | 1959 | 62 | F | 159 | 89.2 |
| | 012 | 1964 | 56 | F | 163 | 56.5 |
| | 013 | 1971 | 49 | M | 173 | 89.5 |
| | 014 | 1969 | 51 | M | 169 | 75.8 |
| Placebo | 003 | 1956 | 70 | F | 188 | 75.5 |
| | 005 | 1978 | 42 | M | 152 | 45.6 |

### TITLES AND FOOTNOTES

Titles and footnotes for every type of output are stored in the study's documentation file and they require regular checking by the second line programmer. This part can be verified manually or with usage of R. Basically, outputs in lst/txt formats can be loaded into R with *readLines* function (base package) line by line and filtered for outputting only titles and footnotes. A data frame created in this way can be further validated with the study documentation. Function *filter* is used to filter for $n^{th}$ row of data frame (which contains title) and the last $n^{th}$ rows which contain footnotes. *Str_trim* function from stringr package (part of tidyverse) is used to trim the text in every line.

```
library(stringr)

lst <- data.frame(text = str_trim(readLines("~/path/table_1.lst"))) %>%
  filter(row_number() == 3 | row_number() == n() | row_number() == as.numeric(n()-1))
```

| text |
|---|
| Summary of Baseline Characteristics |
| Note: This is footnote 1. |
| Note: This is footnote 2. |

### PITFALLS

One of the greatest pitfalls while using R for QC purpose is rounding of numbers. The system of rounding in SAS and R is different and it can produce differences while comparing.

| VARIABLE | ..ROWNUMBER.. | BASE | COMPARE |
|---|---|---|---|
| col2 | 8 | 1(**13**%) | 1(**12**%) |

If a value is equal to 5, rounding methods of SAS and R differ. The table below provides outputs from SAS and R performed on the same numbers. SAS rounds up when R rounds to even numbers. It causes differences while comparing which can be explained in documentation by the second line programmer or another approach can be chosen, for instance, a specific function for rounding [1].

*Table 1. Comparison of SAS and R rounding.*

| Number | SAS | R |
|---|---|---|
| 1.5 | 2 | 2 |
| 2.5 | 3 | 2 |
| 3.5 | 4 | 4 |
| 4.5 | 5 | 4 |
| 12.5 | 13 | 12 |

The next issue which should be mentioned is comparing classes of objects in R. Using *diffdf* function not only examines values between two data frames but it also verifies attributes. After loading datasets into R with haven package, labels for variables are stored in attribute named “label”. Validating TLFs also has to include verifying attributes of variables.

```
There are columns in BASE and COMPARE with differing attributes !!
All rows are shown in table below
                        ================================================
                         VARIABLE  ATTR_NAME  VALUES.BASE  VALUES.COMP
                        ------------------------------------------------
                            TRT01P     label     Treatment       XXX
                        ------------------------------------------------
```

## CONCLUSION

R is an open-source platform with a large number of packages which can be used for data manipulations and further for quality control of deliverables, but there is some uncertainty around using it for validation in regulated environments such as pharmaceutical industry. Tidyverse and pharmaverse are among the most popular and reliable sets of packages which can be utilized by programmers. It should be mentioned that validation of outputs prepared in SAS can be problematic with R due to different data formats and rounding of numbers, but it only improves Quality Control procedure and increases the importance of independent programming along with programmers’ knowledge. Programmers have to remember that copying and looking into the other side programmer’s codes is forbidden. Using a different type of statistical software by second line programmers minimizes this risk.

Writing codes in a regulated environment influences the development of new medicines. Usage of another type of statistical platform for replicating results of statistical analysis strengthens the validation process and confidence. It justifies that numbers produced by SAS are proper and can be used for study interpretation.

## REFERENCES


Tidyverse: https://www.tidyverse.org/

Pharmaverse: https://pharmaverse.org/

Diffdf: https://cran.r-project.org/web/packages/diffdf/diffdf.pdf

Imager: https://dahtah.github.io/imager/imager.html

*R: Regulatory Compliance and Validation Issues A Guidance Document for the Use of R in Regulated Clinical Trial Environments*, 2021, https://www.r-project.org/doc/R-FDA.pdf

*Is R language reliable and efficient tool for programming SAS datasets or just art for art’s sake?,* Piotr Podlewski, https://www.lexjansen.com/phuse-us/2020/dh/DH14.pdf

*SAS® Programming in the Pharmaceutical Industry, Second Edition,* Jack Shostak, SAS Institute Inc., 2014

[1] *Generating TFLs in R - Challenges and Successes compared to SAS*, Amol Waykar, Kevin Kramer, Kalyani Komarasetti, Andrew Miskell https://www.lexjansen.com/phuse-us/2020/ct/CT05.pdf


## CONTACT INFORMATION

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


Author Name: Monika Ludwinska

Company: Intego Group LLC

Address: Grzybowska 78

City / Postcode: Warsaw 00-844

Work Phone: +48 222 500 032 ext. 2583

Email: monika.ludwinska@intego-group.com

Web: https://intego-group.com/