PharmaSUG 2021 - Paper AP-156

# "Packages" in R for Clinical Data Analysis, Let's Demystify

Nagalakshmi Kudipudi, Quartesian Research Private Limited

## ABSTRACT

R is a high-level, interpreted and an open-source programming language-based software which is known for statistical analysis and graphical reporting. R offers a Plethora of 'Packages' for data import, wrangling, visualization and analysis. The packages are extensions to the R statistical programming language and fundamental units of reproducible R code. They include reusable R functions, the documentation that describes how to use them, sample data in a standardized collection format that can be installed by users of R, typically via a centralized software repository such as CRAN (the Comprehensive R Archive Network), GitHub.

GitHub, Inc. provides access control and several collaboration features such as bug tracking, feature requests, task management, continuous integration and Wikis for every project. Its more advanced professional and enterprise services are commercial. Free GitHub accounts are commonly used to host open-source projects.

A group of packages called the Tidyverse, which can be considered a "dialect of the R language", is increasingly popular in the R ecosystem. Packages like data.table, dplyr, tidyr and knitr etc. containing functions which can efficiently manipulate complex data sets and create TLF's.

The scope of this paper is to present the RStudio program that demonstrate the relevant 'Packages' in R for generating the summary tables of Clinical Data. Healthcare represents an exponential growth in the volume of data collected in ever more elaborate Clinical Trails. To meet these demands, Clinical Data Scientists are increasingly choosing open-source solutions to leverage the active open-source communities of experienced developers and statisticians for Clinical Data Analysis

## INTRODUCTION

The Pharmaceutical Industry is seeking better alternative technologies and tools which are sustainable and can provide optimal solutions to address Industry challenges effectively. Are our current tools outdated? Do we have an alternate solution for SAS (to avoid high license and maintenance cost)? Is Industry ready for R or Python or another tool? We need efficient Data Science technologies and tools that can help us to manage Data Lake?? (Example: Big Data, Real World Data) to process it faster and accurately. Efficiency in Data Analysis results in greater insights about data and can help improve decision-making across Drug Development.

Innovation is needed to move away from any traditional inefficient process/tools toward efficient, simple, easy to implement, reliable and cost-effective solutions. Collaboration across Industry stakeholders is needed to develop better technology ecosystems and agree on Validation, and Regulatory benchmarks. It is vital that we prepare our workforce with necessary skillsets for future needs.

This paper highlights why we have to learn R, current status, observed challenges of R and I will share my experience in exploring the potential of R/RStudio® packages in handling TFL tasks which usually done by SAS® by generating descriptive summary table of clinical data and will give brief description of the packages and important features comparison between SAS and R.

## WHY LEARN R?

R is an open-source language. It is maintained by a community of active users and you can avail R for free i.e. Free Installation. You can modify various functions in R and make your own packages. Since R is issued under the **General Public License (GNU)**, there are no restrictions on its usage.

It has become most popular and Hottest Trend setter programming language in the world of Data Science. R is a platform-independent language. It is a cross-platform programming language, meaning that it can be run quite easily on Windows, Linux, and Mac. It's having large community of users, there are approximately 2 million users worldwide for R.

## DEEP DIVE INTO R:

R has 3 decades of legacy. R was created by Robert Gentleman and Ross Ihaka at the University of Auckland, New Zealand (Similar to the S language) in year 1993. Since 1997, there has been an R Core group with write access to R source code. As an open-source software, R receives huge support from the Community. Source code availability provides superior and thorough documentation.

R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible. One of R's strength is ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. R has developed rapidly and has extended by a large collection of packages.

The latest CRAN (Comprehensive R Archive Network) repository has around 17320+ R packages. Various R packages exists on CRAN (Comprehensive R Archive Network) consisting of several functions. All functions need not to be understood in order to perform analysis tasks. You can understand basic and essential R functions to get the grip on software.

**Current Trends of R in Pharmaceuticals:** Looking at the current industry trends, R usage is less than 10% in activities related to Pharmaceutical Regulatory Submissions at this juncture. However, R is extensively used in public health projects, healthcare economics, exploratory/scientific analysis, trend identification, generation of Plots/Graphs, specific statistical analysis and machine learning.

R Validation hub is a Cross Industry Initiative. The mission is to enable the use of R by the Bio-Pharmaceutical Industry in a regulatory setting, where the output may be used in submissions to regulatory agencies.

The R Validation Hub comprises of participants from across the pharmaceutical industry (Abbvie, Amgen, Astellas, Bayer, Boehringer-Ingelheim, Celgene, Eli Lilly, FDA, Genentech, Gilead, GSK, Johnson & Johnson, Merck, Novartis, Novo Nordisk, Pfizer, Roche, RStudio, Sanofi, Teva Pharmaceutical Industries Ltd, and many more). Participants contribute to the effort through regular group meetings, as well as the various workstreams that make up the project.

The focus of this group is on designing a framework that assesses the quality of an R package (Contributed by volunteers) and to create a repository of "accepted" packages.

### What Is A Package?

R packages are a collection of R functions. They are stored under a directory called "library" in the R environment. By default, R installs a set of packages during installation. More packages are added later when they are needed for some specific purpose. When we start the R console, only the default packages are available by default. Other packages which are already installed have to be loaded explicitly to be used by the R program that is going to use them. The basic information about a package is provided in the DESCRIPTION file, where you can find out what the package does, who the author is, what version the documentation belongs to, the date, the type of license its use, and the package dependencies.
Besides finding the DESCRIPTION files such as cran.r-project.org or stat.ethz.ch, we can also access the description file inside R with the command packageDescription("package"), via the documentation of the package help(package = "package"), or online in the repository of the package.

For example, for the “stats” package, these ways will be:
packageDescription("stats")
help(package = "stats")

## WHAT ARE REPOSITORIES?

A repository is a place where packages are located so you can install them from it. Three of the most popular repositories for R packages are:

**CRAN:** (the Comprehensive R Archive Network), the official repository, it is a network of ftp and web servers maintained by the R community around the world. The R foundation coordinates it, and for a package to be published here, it needs to pass several tests that ensure the package is following CRAN policies. We can find more details at https://cran.r-project.org/web/packages/policies.html.

**Bioconductor:** this is a topic specific repository, intended for open sources software for bioinformatics. As CRAN, it has its own submission and review processes, and its community is very active having several conferences and meetings per year.

**Github:** although this is not R specific, Github is probably the most popular repository for open source projects. Its popularity comes from the unlimited space for open source, the integration with git, a version control software, and its ease to share and collaborate with others.

## BRIEF DESCRIPTION OF SOME COMMON “R” PACKAGES WHICH ARE USED FOR TLF GENERATION:

**Package ‘haven’:** Import foreign statistical formats into R via the embedded 'ReadStat' C library, Import and Export 'SPSS', 'Stata' and 'SAS' Files. Package contains functions like (as_factor, labelled, print_labels, read_dta, read_sas, read_xpt, etc.

**read_sas:** Read and write SAS files. read_sas() supports both sas7bdat files and the accompanying sas7bcat files that SAS uses to record value labels. write_sas() is currently experimental and only works for limited datasets.

Below is the example with code snippet shows installing the haven package to import the ADLB SAS dataset and to view the ADLB dataset in Rstudio environment.

```
install.packages("haven")
library(haven)
#importing adlb dataset from study library
adlb <- read_sas("C:/Users/nagalakshmi.kudipudi/Documents/R/Data/adlb.sas7bdat", NULL)
view(adlb)
```

After Installing the package, they are stored under directory called “Library” in the R environment, from the library we can load it and use it.

View (): Displays the dataset in the R environment.

NULL uses the encoding specified in the file.

Figure 1 shows the typical view of the dataset import options in the Rstudio environment

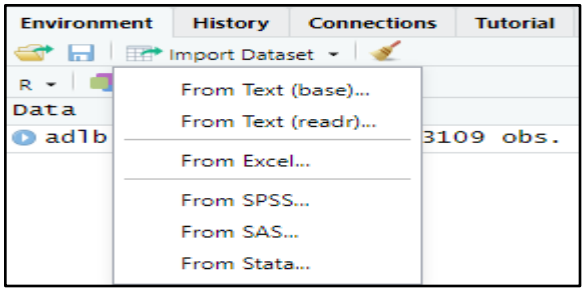
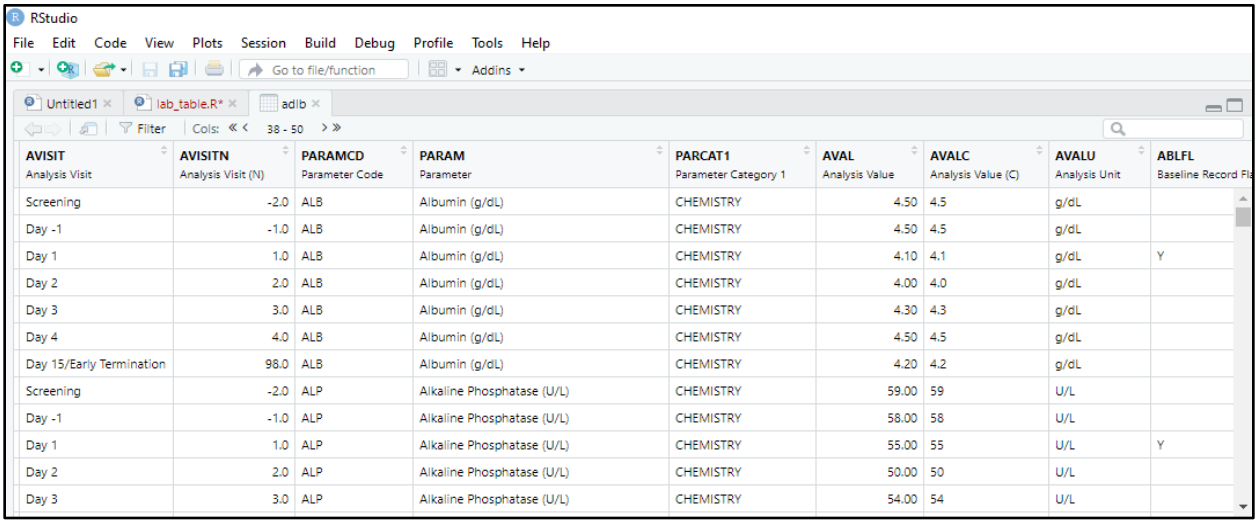



**Figure 1 Data import options**

Figure 2 shows typical view of imported ADLB SAS dataset in the Rstudio layout




| AVISIT<br>Analysis Visit | AVISITN<br>Analysis Visit (N) | PARAMCD<br>Parameter Code | PARAM<br>Parameter | PARCAT1<br>Parameter Category 1 | AVAL<br>Analysis Value | AVALC<br>Analysis Value (C) | AVALU<br>Analysis Unit | ABLFL<br>Baseline Record Fl |
|---|---|---|---|---|---|---|---|---|
| Screening | -2.0 | ALB | Albumin (g/dL) | CHEMISTRY | 4.50 | 4.5 | g/dL | |
| Day -1 | -1.0 | ALB | Albumin (g/dL) | CHEMISTRY | 4.50 | 4.5 | g/dL | |
| Day 1 | 1.0 | ALB | Albumin (g/dL) | CHEMISTRY | 4.10 | 4.1 | g/dL | Y |
| Day 2 | 2.0 | ALB | Albumin (g/dL) | CHEMISTRY | 4.00 | 4.0 | g/dL | |
| Day 3 | 3.0 | ALB | Albumin (g/dL) | CHEMISTRY | 4.30 | 4.3 | g/dL | |
| Day 4 | 4.0 | ALB | Albumin (g/dL) | CHEMISTRY | 4.50 | 4.5 | g/dL | |
| Day 15/Early Termination | 98.0 | ALB | Albumin (g/dL) | CHEMISTRY | 4.20 | 4.2 | g/dL | |
| Screening | -2.0 | ALP | Alkaline Phosphatase (U/L) | CHEMISTRY | 59.00 | 59 | U/L | |
| Day -1 | -1.0 | ALP | Alkaline Phosphatase (U/L) | CHEMISTRY | 58.00 | 58 | U/L | |
| Day 1 | 1.0 | ALP | Alkaline Phosphatase (U/L) | CHEMISTRY | 55.00 | 55 | U/L | Y |
| Day 2 | 2.0 | ALP | Alkaline Phosphatase (U/L) | CHEMISTRY | 50.00 | 50 | U/L | |
| Day 3 | 3.0 | ALP | Alkaline Phosphatase (U/L) | CHEMISTRY | 54.00 | 54 | U/L | |

**Figure 2 Imported SAS Data view in Rstudio layout through haven package**

**Package 'dplyr' :** dplyr is a powerful R-package to transform and summarize tabular data with rows and columns. The package contains a set of functions (or "verbs") that perform common data manipulation operations such as filtering for rows, selecting specific columns, re-ordering rows, adding new columns and summarizing data.

Below table shows Important dplyr verbs to remember.

| dplyr verbs | Description |
|---|---|
| select() | select columns |
| filter() | filter rows |
| arrange() | re-order or arrange rows |
| mutate() | create new columns |
| summarise() | summarise values |
| group_by() | allows for group operations in the "split-apply-combine" concept |

**Package 'magrittr' :** A Forward-Pipe Operator for **R,** Provides a mechanism for chaining commands with a new forward-pipe operator, %>%. This operator will forward a value, or the result of an expression, into the next function call/expression. There is flexible support for the type of right-hand side expressions.

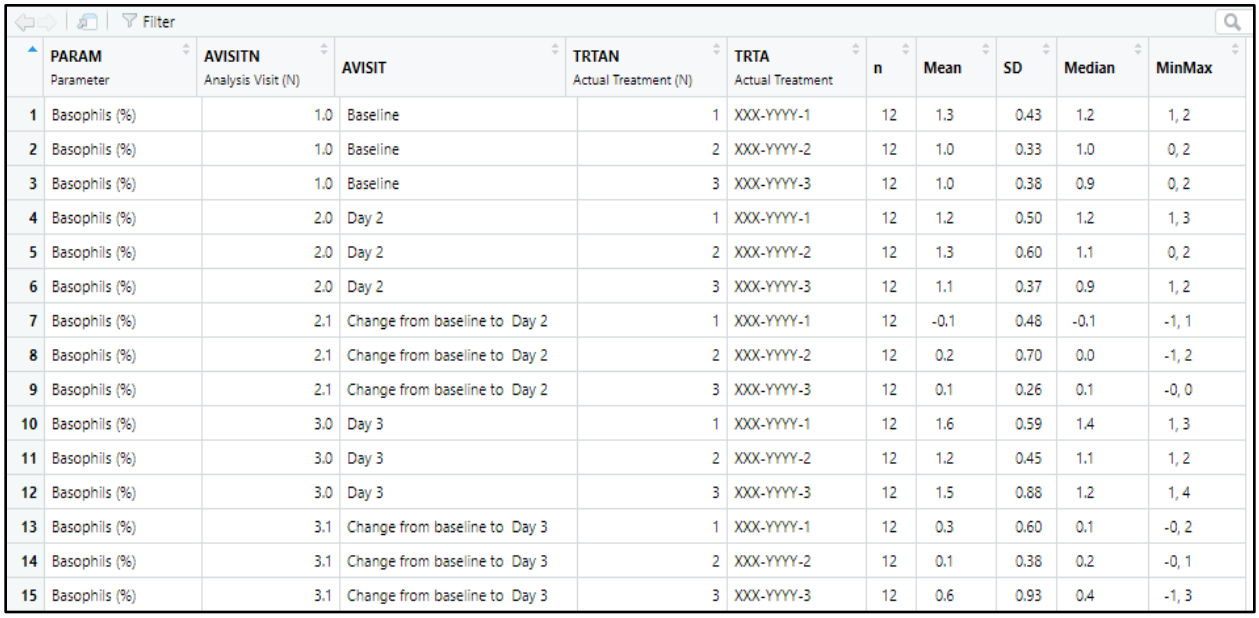
Below is the code snippet example for both 'dplyr', 'magrittr' packages where functions like select(), filter(),rbind(),group_by(), summarise() and forward-pipe operator, %>%. used for sub setting the ADLB dataset as per requirement.

```
LB<-select(LB,"AVISIT","AVAL","PARAM","TRT01A","BASE","CHG","USUBJID","AVISITN","PARCAT1","SAFFL","TRTA","TRTAN","ABLFL")

#Replacing AVISIT with Baseline for ABLFL="Y" records
LB$AVISIT=ifelse(LB$ABLFL=='Y',"Baseline",LB$AVISIT)
#Filtering datasetas per requirement
LB1=LB %>%
   filter(SAFFL == "Y",AVAL!=.,BASE!=.,PARCAT1=='HEMATOLOGY')
LB1$VAR=LB1$AVAL
LB2=LB %>%
   filter(SAFFL=="Y",CHG!=.,PARCAT1=='HEMATOLOGY')
LB2$VAR=LB2$CHG
LB2$AVISIT=paste("Change from baseline to ",LB2$AVISIT)
LB2$AVISITN=LB2$AVISITN+0.1
LB3<-rbind(LB1,LB2)

#Sorting dataest
LB4<-LB3[order(LB3$USUBJID,LB3$PARAM,LB3$AVISITN,LB3$AVISIT,LB3$TRTAN,LB3$TRTA),]
LB4$VAR=as.numeric(LB4$VAR)
```

Below code snippet summarizes dataset and applies the required decimal alignment as per requirement

```
Means <- defmacro(data=data, var=var,
                  expr=
                   {data = LB4 %>%
                    group_by(PARAM,AVISITN,AVISIT,TRTAN,TRTA) %>%
                     summarise(n = sprintf("%4.0f",length(USUBJID)),
                               Mean=sprintf("%6.1f",mean(var)),
                               SD= sprintf("%7.2f",sd(var)),
                               Median=sprintf("%6.1f",median(var)),
                               MinMax=paste(sprintf("%4.0f",min(var)),trimws(sprintf("%4.0f",max(var))),sep=', '))
                              })
Means(data=SUMM,var=VAR)
```

Figure 3 shows the intermediate output dataset generated executing the above code,



| | PARAM<br>Parameter | AVISITN<br>Analysis Visit (N) | AVISIT | TRTAN<br>Actual Treatment (N) | TRTA<br>Actual Treatment | n | Mean | SD | Median | MinMax |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Basophils (%) | 1.0 | Baseline | 1 | XXX-YYYY-1 | 12 | 1.3 | 0.43 | 1.2 | 1, 2 |
| 2 | Basophils (%) | 1.0 | Baseline | 2 | XXX-YYYY-2 | 12 | 1.0 | 0.33 | 1.0 | 0, 2 |
| 3 | Basophils (%) | 1.0 | Baseline | 3 | XXX-YYYY-3 | 12 | 1.0 | 0.38 | 0.9 | 0, 2 |
| 4 | Basophils (%) | 2.0 | Day 2 | 1 | XXX-YYYY-1 | 12 | 1.2 | 0.50 | 1.2 | 1, 3 |
| 5 | Basophils (%) | 2.0 | Day 2 | 2 | XXX-YYYY-2 | 12 | 1.3 | 0.60 | 1.1 | 0, 2 |
| 6 | Basophils (%) | 2.0 | Day 2 | 3 | XXX-YYYY-3 | 12 | 1.1 | 0.37 | 0.9 | 1, 2 |
| 7 | Basophils (%) | 2.1 | Change from baseline to Day 2 | 1 | XXX-YYYY-1 | 12 | -0.1 | 0.48 | -0.1 | -1, 1 |
| 8 | Basophils (%) | 2.1 | Change from baseline to Day 2 | 2 | XXX-YYYY-2 | 12 | 0.2 | 0.70 | 0.0 | -1, 2 |
| 9 | Basophils (%) | 2.1 | Change from baseline to Day 2 | 3 | XXX-YYYY-3 | 12 | 0.1 | 0.26 | 0.1 | -0, 0 |
| 10 | Basophils (%) | 3.0 | Day 3 | 1 | XXX-YYYY-1 | 12 | 1.6 | 0.59 | 1.4 | 1, 3 |
| 11 | Basophils (%) | 3.0 | Day 3 | 2 | XXX-YYYY-2 | 12 | 1.2 | 0.45 | 1.1 | 1, 2 |
| 12 | Basophils (%) | 3.0 | Day 3 | 3 | XXX-YYYY-3 | 12 | 1.5 | 0.88 | 1.2 | 1, 4 |
| 13 | Basophils (%) | 3.1 | Change from baseline to Day 3 | 1 | XXX-YYYY-1 | 12 | 0.3 | 0.60 | 0.1 | -0, 2 |
| 14 | Basophils (%) | 3.1 | Change from baseline to Day 3 | 2 | XXX-YYYY-2 | 12 | 0.1 | 0.38 | 0.2 | -0, 1 |
| 15 | Basophils (%) | 3.1 | Change from baseline to Day 3 | 3 | XXX-YYYY-3 | 12 | 0.6 | 0.93 | 0.4 | -1, 3 |

**Figure 3**

**Package 'reshape2' :** Flexibly restructure and aggregate data using just two functions: melt and 'dcast' (or 'acast')., Data Reshaping or transposing in R is something like arranged rows and columns in our own way to use it as per our requirements, mostly data is taken as a data frame format in R to do data processing using functions like melt and 'dcast' (or 'acast')., etc. In this process, you reshape or re-organize/transpose the data into rows and columns.

Below is the code snippet for reshaping or transposing the statistics variables (i.e. wide data into vertical format) as per the requirement.

```
library(reshape2)
OUT1=reshape(data=SUMM, idvar =c("PARAM","AVISITN","AVISIT"),
             varying=c("n","Mean","SD","Median","MinMax"),
             v.name=c("Stat"),
             times =c("1n","2Mean","3SD","4Median","5Min, Max"),
             new.row.names=1:10000,
             direction="long")
TAB=arrange(OUT1,PARAM,TRTA,as.numeric(AVISITN))
TAB1=dcast(setDT(TAB),PARAM+AVISITN+AVISIT+time~TRTAN,value.var="Stat")
TAB1$time=ifelse(TAB1$time=="1n","n",TAB1$time)
TAB1$time=ifelse(TAB1$time=="2Mean","Mean",TAB1$time)
TAB1$time=ifelse(TAB1$time=="3SD","SD",TAB1$time)
TAB1$time=ifelse(TAB1$time=="4Median","Median",TAB1$time)
TAB1$time=ifelse(TAB1$time=="5Min, Max","Min, Max",TAB1$time)
TAB1<-rename(TAB1,c(time="STATISTICS"))
```

Figure 4 shows the final data set generated by executing above code which is used to create final RTF or PDF output.

Filter

| | PARAM | AVISITN | AVISIT | STATISTICS | 1 | 2 | 3 |
|---|---|---|---|---|---|---|---|
| 1 | Basophils (%) | 1.0 | Baseline | n | 12 | 12 | 12 |
| 2 | Basophils (%) | 1.0 | Baseline | Mean | 1.3 | 1.0 | 1.0 |
| 3 | Basophils (%) | 1.0 | Baseline | SD | 0.43 | 0.33 | 0.38 |
| 4 | Basophils (%) | 1.0 | Baseline | Median | 1.2 | 1.0 | 0.9 |
| 5 | Basophils (%) | 1.0 | Baseline | Min, Max | 1, 2 | 0, 2 | 0, 2 |
| 6 | Basophils (%) | 2.0 | Day 2 | n | 12 | 12 | 12 |
| 7 | Basophils (%) | 2.0 | Day 2 | Mean | 1.2 | 1.3 | 1.1 |
| 8 | Basophils (%) | 2.0 | Day 2 | SD | 0.50 | 0.60 | 0.37 |
| 9 | Basophils (%) | 2.0 | Day 2 | Median | 1.2 | 1.1 | 0.9 |
| 10 | Basophils (%) | 2.0 | Day 2 | Min, Max | 1, 3 | 0, 2 | 1, 2 |
| 11 | Basophils (%) | 2.1 | Change from baseline to Day 2 | n | 12 | 12 | 12 |
| 12 | Basophils (%) | 2.1 | Change from baseline to Day 2 | Mean | -0.1 | 0.2 | 0.1 |
| 13 | Basophils (%) | 2.1 | Change from baseline to Day 2 | SD | 0.48 | 0.70 | 0.26 |
| 14 | Basophils (%) | 2.1 | Change from baseline to Day 2 | Median | -0.1 | 0.0 | 0.1 |
| 15 | Basophils (%) | 2.1 | Change from baseline to Day 2 | Min, Max | -1, 1 | -1, 2 | -0, 0 |

Figure 4

We can use RTF/ r2rtf package to create the final RTF output, you can find more details on this package here.

## CONCLUSION:

Comparison between SAS and R

The use of R as software of choice within pharmaceutical companies and Contract Research Organizations (CROs) is something that many doubted to see during their lifespan, however things are rapidly moving even in the pharmaceutical industry. Nevertheless, many misconceptions about R still exist, not least whether it's a tool suitable for creating submission-ready deliverables such as TLFs. In this paper we have seen that R can be an extremely powerful tool by summarizing the analysis lab dataset by using some common R packages already available and that by leveraging its high flexibility, it is possible to obtain high- quality results with comparable efficiency and quality to standard SAS code.

FDA does not require use of any specific software for statistical analyses, and statistical software is not explicitly discussed in Title 21 of the Code of Federal Regulations [e.g., in 21CFR part 11]. However, the software package(s) used for statistical analyses should be fully documented in the submission, including version and build identification. The FDA's Statistical Software Clarifying Statement declares that any suitable software can be used in a regulatory submission.

Since last few decades clinical industry is evolving, and it requires standardization all over the world. SAS played major role in analysis of data for submission. SAS is obvious leader for analysis in health care industry as it is recognized and accepted by regulatory bodies across the world. Industry folks find it easy to learn but expensive one when it comes to individual use.

At more than 20 years old, R is fairly mature and growing in popularity. R is used by a growing number of data analysts inside corporations and academia, whether being used to set ad prices, find new drugs more quickly or finetune financial models

R is free for anyone to use and modify so statisticians, engineers and data scientists can improve the software's code or write variations for specific tasks.

## REFERENCES


https://cran.r-project.org/web/packages/haven/index.html

https://cran.r-project.org/web/packages/dplyr/index.htm

www.coursera.org

https://cran.r-project.org/web/packages/reshape2/index.html


## ACKNOWLEDGMENTS


I would like to acknowledge my colleague Abhijit Bhasker Shinde and senior manager Kannan Murugaiyan who have helped me in writing and reviewing this paper.


## CONTACT INFORMATION

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


Nagalakshmi Kudipudi
Quartesian Research Private Limited, Hyderabad ,India
Nagalakshmi.kudipudi@quartesian.com