Paper PD05

R-Guru Resource Hub for Rapid R Learning

Sunil Gupta, Gupta Programming, Simi Valley, USA

ABSTRACT

Are you behind in leveraging Pharmaverse R package releases for SDTMs, ADaMs and TLFs? Do you understand the benefits of learning and applying R instead of only programming in SAS? Do you know how to apply basic R syntax using common and validated R packages by R Studio and the pharma industry? Are you interested in adding R to your programming skill set so that you have multiple methods for data input, management and analysis?

Learning R for professional development is a new trend for early adaptor corporations and SAS programmers. Because R is more an object-oriented and syntax direct language, programmers can now write concise code for targeted tasks. This empowers programmers with a competitive advantage for advancing their career.

INTRODUCTION

R-Guru.com is now a resource hub with great content to help SAS programmers make the transition to R! Based on my current R consulting projects, I created R-Guru.com and cheat sheet to contain practical and real-world R examples for data input, variable and data frame process and reporting. In the R-Guru cheat sheet, the section on compare and contrast common R functions is an overview of tasks and at least two methods to complete each task. As with SAS, there are multiple methods to perform the same task. Also included in the R-Guru cheat sheet is the section on debugging R. From reviewing the error type, I propose several potential solutions.

Learn R functions from common packages, best practices, Pharmaverse and popular books and blogs. Below is the outline for this paper.

- Avoid the Steep Learning Curve
- Apply R Best Practices
- Leverage R Cheat Sheets
- Learn Pharmaverse R packages
- Read on-line R books and blogs

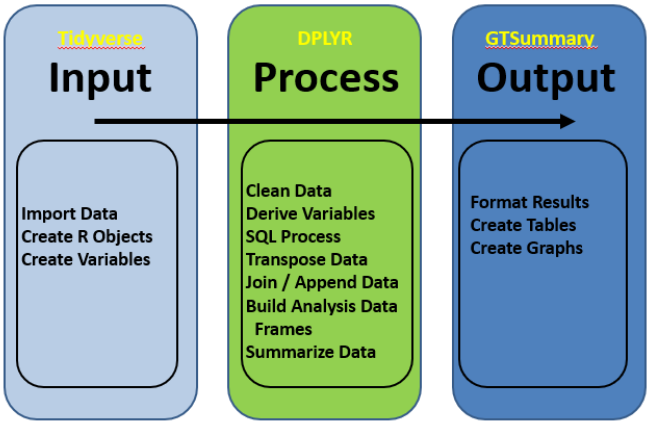
AVOID THE STEEP LEARNING CURVE

R is technical language that requires remembering R syntax. R is an interpreted language so your R code is directly executed instead of being compiled into an object language like SAS. R is a true, object-based language, powerful and concise code based on special characters ([], $).

To avoid the steep learning curve, focus on applying simple hands-on exercises from common R packages such as tidyverse and DPLYR. For R programming only, stay focused on common R functions and not get distracted with matrices or statistical modeling. Basic exercises help build understanding and confidence. In addition, a mentor helps to address challenges and questions since debugging is very difficult.

R has packages and functions for data input, management, analysis and reporting.

R Process: Data Input to Statistical Analysis



R is similar to SAS in that the SAS Display Manager is R Studio. R also has character, numeric and date variables. SAS datasets are data frames in R. Finally, SAS do loops are loop functions in R. Running simple examples and exercises are some of the best methods for learning R.

Basic R Examples with comments

Data Management Operations:

```
# Create Data Frame
mydata <- data.frame(
 class = c("1st", "2nd", "3rd", "Crew"),
 n = c(325, 285, 706, 885),
 prop = c(14.8, 12.9, 32.1, 40.2)
)
mydata

# Keeping Variables
test_df2=mydata[c('class', 'n')]
test_df2
```

```
# Dropping Variables
test_df3= subset(mydata, select = -c(class))
test_df3

test_df4= mydata[-c(3)]
test_df4

mydata$myvar <- NULL
# subset(x, subset, select, drop = FALSE, ...)
x – data frame
subset – Subset expression
select – Keep variables
```

R Exercises for each type of Task

Data Management Operations Exercises (*Next Section*)

1. Create mydata1 data frame from dropping gender1 variable in mydataframe.
2. Create mydata2 data frame from keeping gender and age variables in mydataframe.
3. Create mydata2b data frame by creating new variable newvar as if age > 50 then 'Above 50' else '50 or Below'. (cut(), case_when(), mutate() with case_when(), mutate())
4. In mydataframe, rename variable gender to sex.
5. Replace NA values with Zeros.

Just like SAS, R is a programming language that is built on process, order, logic and comments.

SDTM/ADaM Datasets using R

R- Dplyr Syntax

Subject level derivation -Sample code:

```
adsl <-   dm %>%      # read %>% as "and then"
          select(studyid, subjid, age, sex, height, weight, race, scrfl) %>%
          mutate(bmi = (weight*703)/height^2 ) %>%
          filter(scrfl == "Y") %>%
          select(-scrfl) %>%
          arrange(studyid, subjid)
```

Piping, '%>%', is unique to R. Piping enables programmers to concatenate R functions to do multiple tasks: Select, Mutate, Filter, Arrange.

Order

```
6  adsl <- dm %>%    # separate lines per R command help for reading
1  select(studyid, subjid, age, sex, height, weight, race, scrfl) %>%
2  mutate(bmi = (weight*703)/height^2 ) %>%
3  filter(scrfl == "Y") %>%
4  select(-scrfl) %>%
5  arrange(studyid, subjid)
```

With %>%, several R commands execute together which is similar to SAS Procedures.

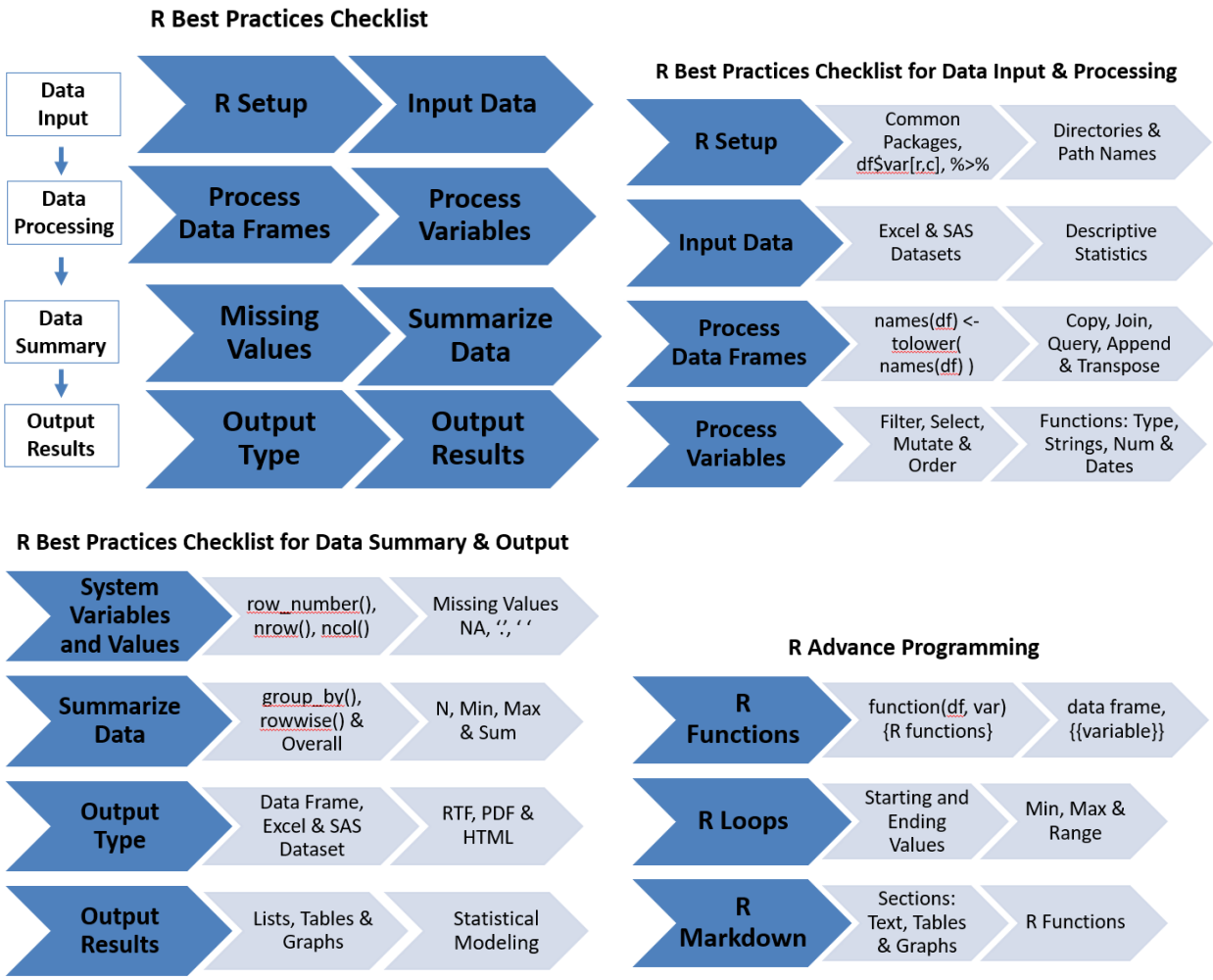
SAS dataset options are direct variable and record references in R. Below are some examples.

```
print(df1[df1$vr1 == 'male', c('vr1', 'vr2')])   # print vr1 & vr2 for males

data$vr2 <- ifelse(data$vr1 >= 4, 1, 0)  # derive vr2 by condition, True, False

class[class$Age>=mean(class$Age),] # filter age >= mean age, all class vars

data$agecat[num_range(40 – data$age – 60)] # derive agecat as 40 - 60
```

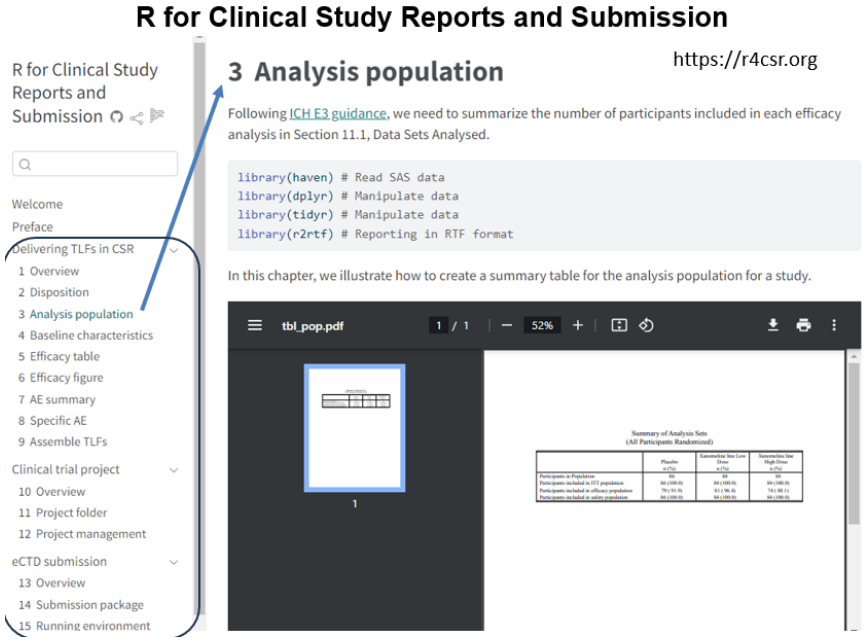
```
df2 <- df1[ row conditions / #, column conditions / # ]
df2$vr2 <- R( df1$vr1 condition ) # vr2 assignment
df2$vr2[ df1$vr1 condition ] <- constant # vr2 assignment
df[ [ column conditions / # ] ] # returns a vector
```

R BEST PRACTICES

For better results, apply R best practices in data input, process, summary and results. Instead of getting confused by reviewing multiple methods to complete tasks, it is better to learn best methods that cover over 80% of the cases.



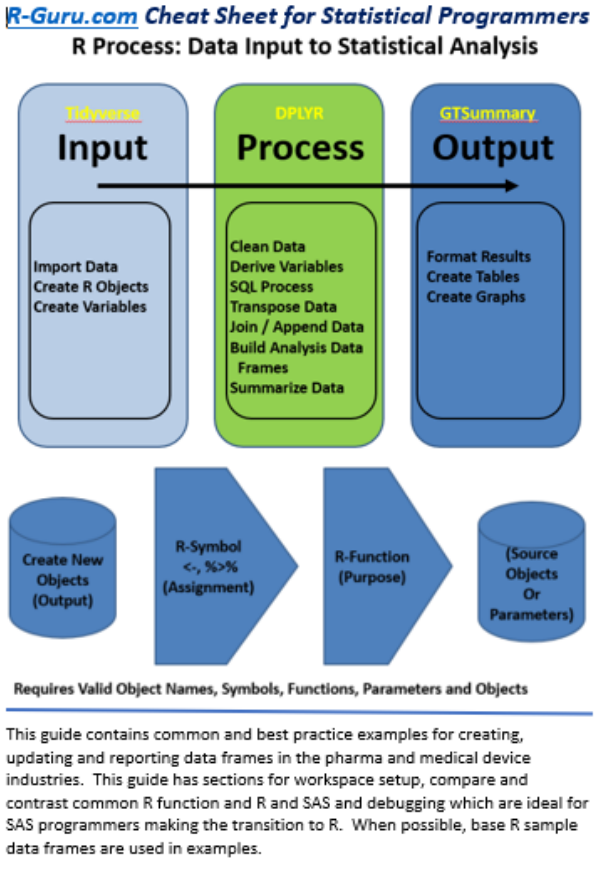
With the collaboration among pharma companies, new R package are developed and are available to the R community such as the R for Clinical Study Reports and Submission for creating tables and lists.



R CHEAT SHEETS

My unique R-Guru cheat sheet is ideal for cutting and pasting R examples into R Studio and then customizing the example. Applying correct and complete R syntax is important to prevent R debugging since this can be a challenge.

For quick review of R syntax for specific tasks, leverage R cheat sheets. Since it is almost impossible to remember R syntax details, accessing R cheat sheets with images help to remind you of the correct R syntax. R-Guru has the collection of the best R cheat sheets that are easy to access when needed.

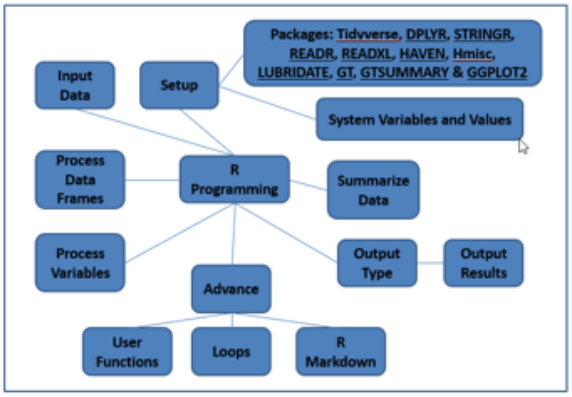


Tidyverse, DPLYR, DATA.TYPE, STRINGR, READR, READXL, HAVEN, Hmisc, arsenal, LUBRIDATE, PARSEDATE, GT, GTSUMMARY & GGPLOT2 are common and validated R packages by RStudio and the Pharma Industry.

Mutate() function has five key features: case_when(), simple expression, summary functions, rowwise(), and group_by()/ungroup() with summary functions.

df# are data frame names & vr# are variable names. Character or numeric variables depend on the function and values. R functions may be nested for multiple tasks.

R-Guru Best Practices Mind Map



While many SAS features can be replicated in R, it is important to be aware of any differences such as how missing data is handled. Since debugging R can be a challenge, a look up table of potential solutions by error type is helpful.

TASK	R	SAS
Language	Interpreter	Compiler and Interpreter
Character Var Length	N/A	length
Rounding 2.5	2 (even number)	3 (up)
Sorting Missing Values	'NA' is last obs unless converted to missing	Missing is first obs
Common Features **Data:** Input (Excel, CSV), Management, Analysis, & Reporting (RTF, PDF) **Var Type:** Character, Numeric and Date Variables **Other:** SQL, Do-Loops	R Studio Data Frames Direct Variable and Record References as.character() , as.numeric() vfmt[df$vr1] R Shiny App	Display Manager Datasets Dataset Options (Keep, Drop, Where) put(), input() proc format

ERROR TYPE	SOLUTIONS
Invalid or Missing Packages, Path names, Libraries not Loaded	Load and confirm packages, path names and libraries
Invalid or Missing Data Frames, Objects or Variables	Confirm correct and existing data frames (instead of matrix), objects and vars, lower case all names since case-sensitive, correct order of tasks (select, filter, etc.) within DPLYR (SQL) functions, apply group_by() before summary functions to prevent overall summaries
Invalid or Missing Functions or Operations	Confirm functions exist and correctly applied, confirm variable and function types are consistent
Invalid or Missing Parameters and Options	Confirm correct function usage, case-sensitive, cut/paste working example
Invalid or Missing Data or Format	Confirm data import is correct, lower case data since case-sensitive, remove extra spaces before and after data values, confirm correct date format, apply factors to assign invalid data as NA, data by descriptive stats, freq counts, min, max, etc.
Invalid Logic	Confirm process logic flow, test and view inputs and outputs of each function

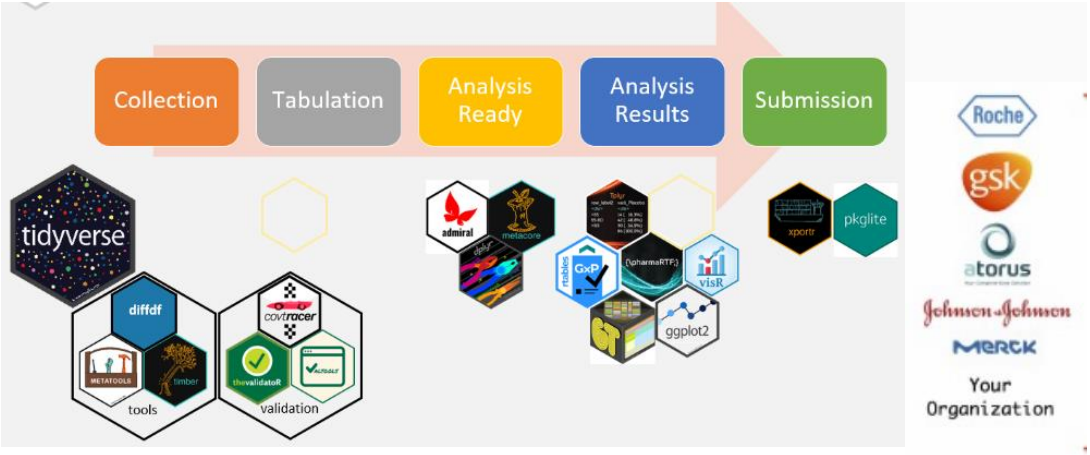
As in SAS, there are multiple methods in R to accomplish the same task. The below table helps to compare similar R functions by task.

TASK	METHOD 1	METHOD 2
Query, Add Variables	mutate(dose2 = (dose*2)) cbind(df1, vr1=1, vr2='Drug A')	df1[df1$vr1 == 'male', c('vr1', 'vr2')] # df options
Add Variables by Conditions	case_when(grep("Yes", vr1) ~ 'Yes')	ifelse(data$vr1 >= 4, 1, 0) , if_else()
Add Summary Variables (Overall)	summarize(mean_mpg = mean(mpg, na.rm = TRUE)) mutate(vr3 = mean(vr2, .1))	summarise_at(vars(mpg, wt), list(m=mean, sd=sd), na.rm=TRUE) apply(mtcars, 2, mean)
Group By Vars	group_by(vr)	ungroup() # best practices to prevent subsequent group processing, best used with mutate() to keep all variables
Variable Type Conversion	as.character(vr1)	as.numeric(vr1) as.Date("2021-01-25")
Recode Values	vfmt <- c("M"="Male", "MALE"="Male", "F"="Female", "FEMALE"= "Female") df$vr2 <- vfmt[df$vr1]	recode(vr1, 'val1'='val1a', 'val2'='val2a') recode(vr1, !!!vfmt$vr1))

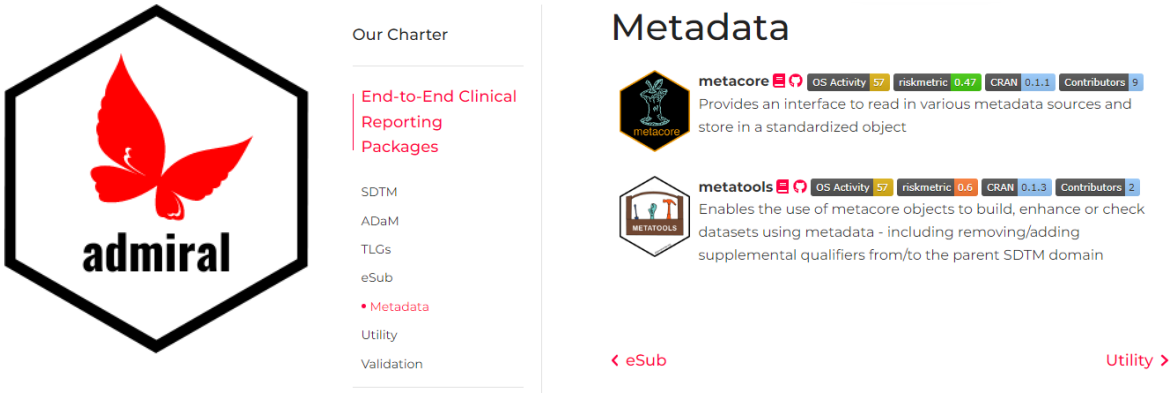
PHARMAVERSE PACKAGES

For the first time in pharma history, there is collaboration between pharma companies and industry to build Pharmaverse R packages.

To streamline creating SDTMs and ADaMs with R program templates, access Pharmaverse R packages. With the growth of these packages, more pharma companies will expect programmers to become familiar with them when programming.

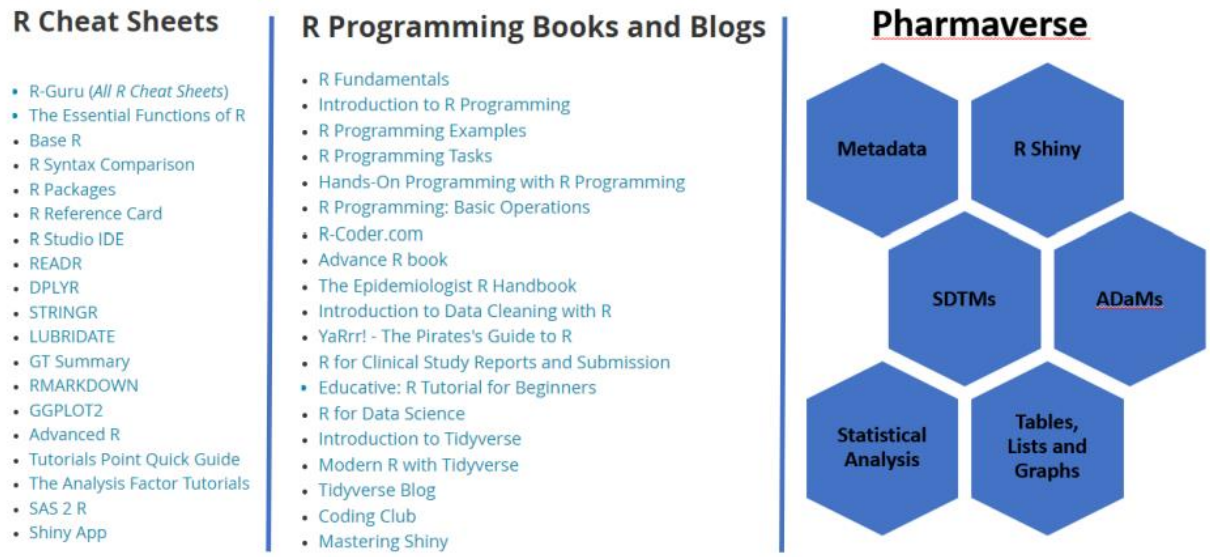


Soon Pharmaverse R packages will be *industry standards:* Metacore, Metatools, SDTMChcks, Datacutr, Admiral, Teal.



R BOOKS AND BLOGS

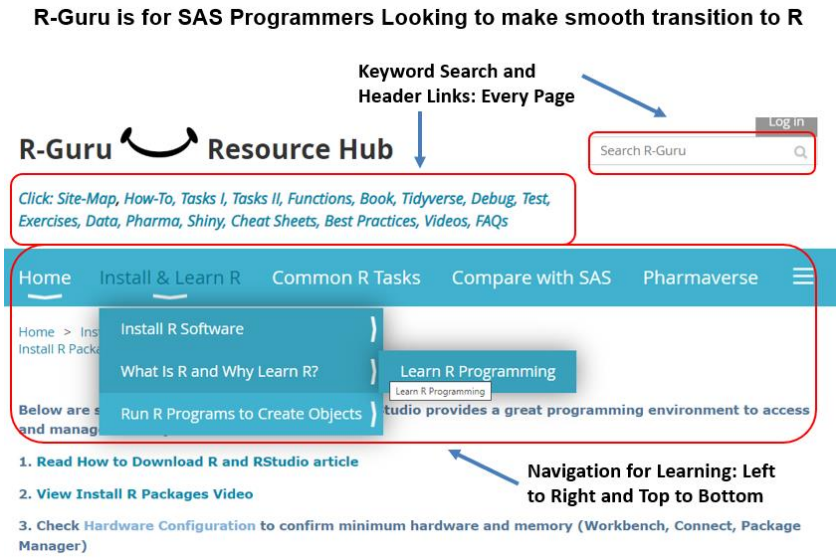
For deep dive learning of R topics, read R books and blogs. The collection of 20 R books and blogs give you a full variety of R program applications with examples.



CONCLUSION

In conclusion, R-Guru was built to help SAS programmers make a smooth transition to R by using structured learning by examples and without having the steep learning curve. From importing data into R to data management and analysis, SAS programmers can now learn R within weeks instead of months. In addition, SAS programmers can have a better understanding of the R syntax by the step-by-step process and compare and contrast learning methods. Mentoring is also an important component to help guide and address R challenges.

The R-Guru Resource Hub is easy to navigate as well as search for R answers and solutions. Whether you are a beginner or an advanced R programmer, R-Guru has resources to fit your needs. With the growth of Pharmaverse, R-Guru will continue to add content and summary tips to keep R programmers current.



ACKNOWLEDGMENTS

We would like to thank Abraham Pulavarti, Supriya Moghe and Paula Jackson for accepting my paper in the professional development section.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:
Sunil Gupta, R Mentor and Trainer, Submission SME
Gupta Programming
213 Goldenwood Circle
Simi Valley, CA 93065
GuptaProgramming@gmail.com
Founder of R-Guru.com and SASSavvy.com