

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

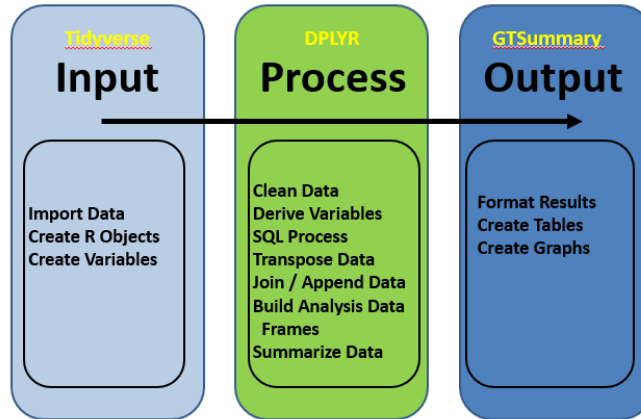
AVOID THE STEEP LEARNING CURVE

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
```

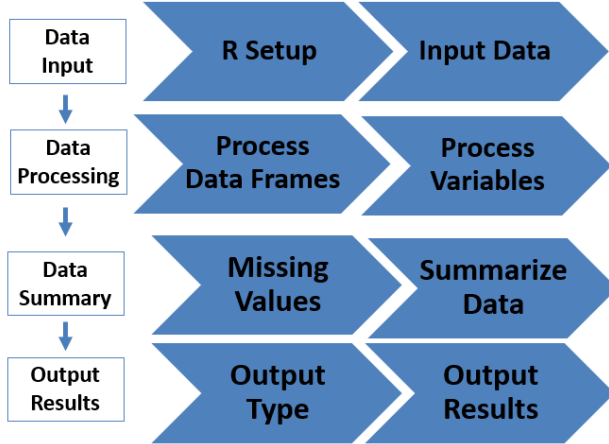
```
data$agecat[num_range(40 – data$age – 60)] # derive agecat as 40 - 60
```

```
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
```

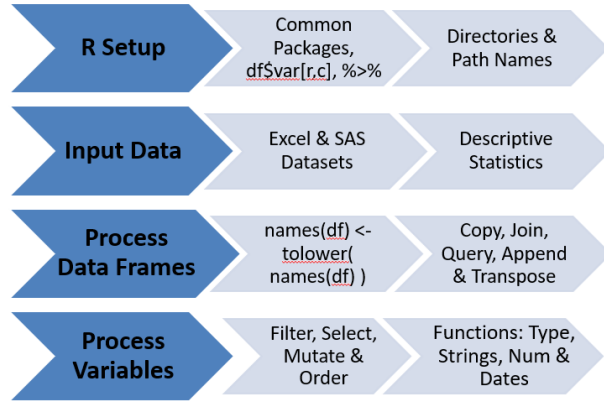
R BEST PRACTICES

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.

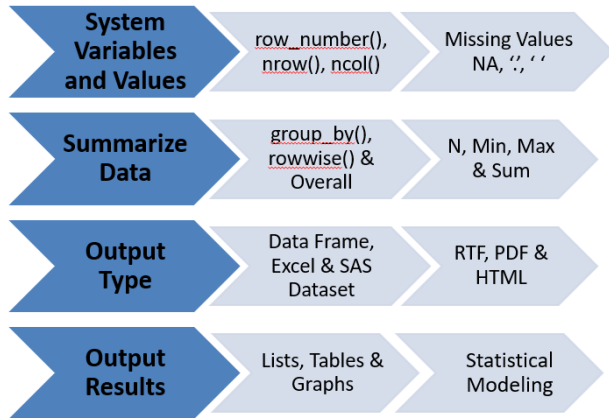
R Best Practices Checklist



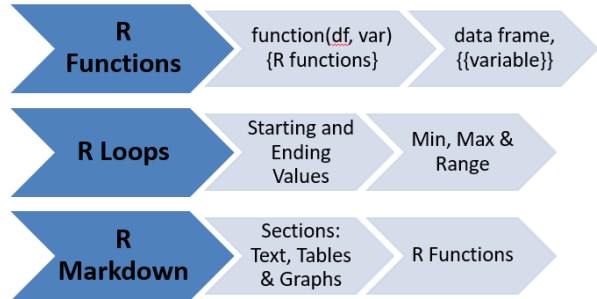
R Best Practices Checklist for Data Input & Processing



R Best Practices Checklist for Data Summary & Output



R Advance Programming



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 for Clinical Study Reports and Submission

R for Clinical Study Reports and Submission <https://r4csr.org>

Welcome

Preface

Delivering TLFs in CSR

- 1 Overview
- 2 Disposition
- 3 Analysis population
- 4 Baseline characteristics
- 5 Efficacy table
- 6 Efficacy figure
- 7 AE summary
- 8 Specific AE
- 9 Assemble TLFs

Clinical trial project

- 10 Overview
- 11 Project folder
- 12 Project management

eCTD submission

- 13 Overview
- 14 Submission package
- 15 Running environment

3 Analysis population <https://r4csr.org>

Following [ICH E3 guidance](#), we need to summarize the number of participants included in each efficacy analysis in Section 11.1, Data Sets Analysed.

```
library(haven) # Read SAS data
library(dplyr) # Manipulate data
library(tidyr) # Manipulate data
library(r2rtf) # Reporting in RTF format
```

In this chapter, we illustrate how to create a summary table for the analysis population for a study.

tbl_pop.pdf 1 / 1 52%

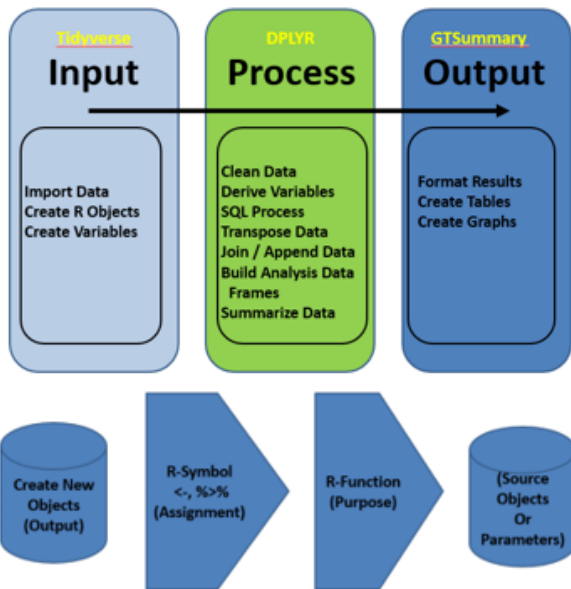
Summary of Analysis Sets (All Participants Randomized)			
	Placebo	Intermediate Dose Group	High Dose
Participants in Population	84	84	84
Participants included in ITT population	84 (100.0%)	84 (100.0%)	84 (100.0%)
Participants included in efficacy population	79 (94.0%)	81 (96.4%)	81 (96.4%)

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.

R-Guru.com Cheat Sheet for Statistical Programmers R Process: Data Input to Statistical Analysis



Requires Valid Object Names, Symbols, Functions, Parameters and Objects

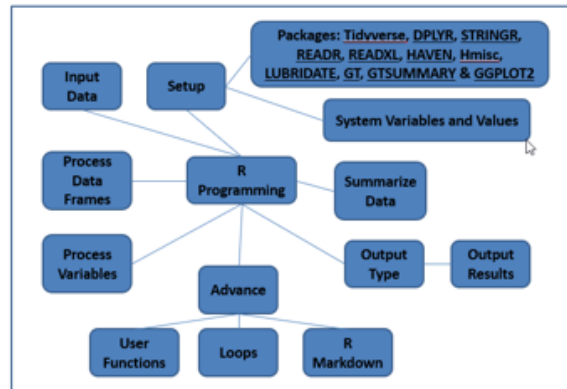
This guide contains common and best practice examples for creating, updating and reporting data frames in the pharma and medical device industries. This guide has sections for workspace setup, compare and contrast common R function and R and SAS and debugging which are ideal for SAS programmers making the transition to R. When possible, base R sample data frames are used in examples.

[Tidverse](#), [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	R Studio	Display Manager
Data: Input (Excel, CSV), Management, Analysis, & Reporting (RTF, PDF)	Data Frames	Datasets
Var Type: Character, Numeric and Date Variables	Direct Variable and Record References as.character() , as.numeric()	Dataset Options (Keep, Drop, Where) put() , input()
Other: SQL, Do-Loops	vfmt[df\$vr1] R Shiny App	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

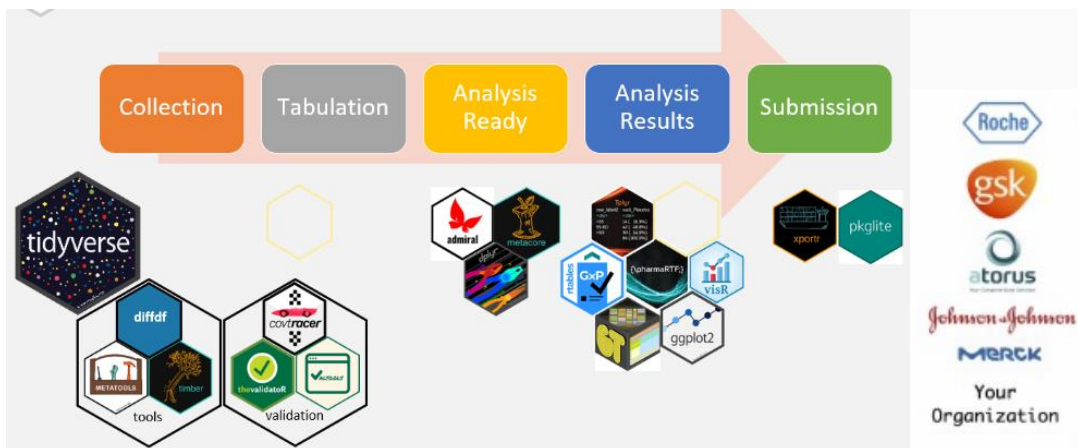
As in SAS, there are multiple methods in R to accomplish the same task. The below table helps to compare similar R functions by task.

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

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.



Our Charter

End-to-End Clinical Reporting Packages

- SDTM
- ADaM
- TLGs
- eSub
- Metadata
- Utility
- Validation

Metadata



metacore 57 0.47 0.1.1 9
Provides an interface to read in various metadata sources and store in a standardized object



metatools 57 0.6 0.1.3 2
Enables the use of metacore objects to build, enhance or check datasets using metadata - including removing/adding supplemental qualifiers from/to the parent SDTM domain

< eSub

Utility >

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.

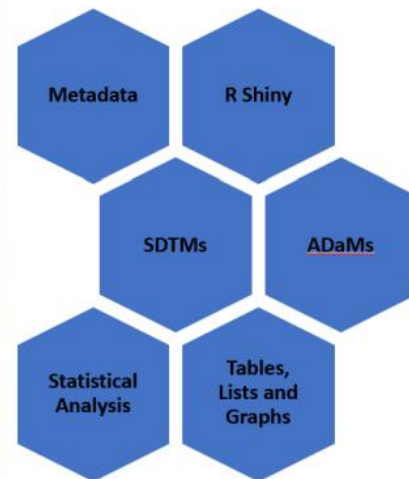
R Cheat Sheets

- [R-Guru \(All R Cheat Sheets\)](#)
- [The Essential Functions of R](#)
- [Base R](#)
- [R Syntax Comparison](#)
- [R Packages](#)
- [R Reference Card](#)
- [R Studio IDE](#)
- [READR](#)
- [DPLYR](#)
- [STRINGR](#)
- [LUBRIDATE](#)
- [GT Summary](#)
- [RMarkdown](#)
- [GGPLOT2](#)
- [Advanced R](#)
- [Tutorials Point Quick Guide](#)
- [The Analysis Factor Tutorials](#)
- [SAS 2 R](#)
- [Shiny App](#)

R Programming Books and Blogs

- [R Fundamentals](#)
- [Introduction to R Programming](#)
- [R Programming Examples](#)
- [R Programming Tasks](#)
- [Hands-On Programming with R Programming](#)
- [R Programming: Basic Operations](#)
- [R-Coder.com](#)
- [Advance R book](#)
- [The Epidemiologist R Handbook](#)
- [Introduction to Data Cleaning with R](#)
- [YaRrr! - The Pirates's Guide to R](#)
- [R for Clinical Study Reports and Submission](#)
- [Educative: R Tutorial for Beginners](#)
- [R for Data Science](#)
- [Introduction to Tidyverse](#)
- [Modern R with Tidyverse](#)
- [Tidyverse Blog](#)
- [Coding Club](#)
- [Mastering Shiny](#)

Pharmaverse



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.

R-Guru is for SAS Programmers Looking to make smooth transition to R

The screenshot shows the R-Guru Resource Hub website. At the top, there is a search bar labeled "Search R-Guru" and a "Log In" button. Below the search bar, there is a navigation menu with links: Home, Install & Learn R, Common R Tasks, Compare with SAS, and Pharmaverse. A red box highlights the search bar and the navigation menu. Below the navigation menu, there is a list of links: Home > Install R Software, What is R and Why Learn R?, Run R Programs to Create Objects, and Learn R Programming. A red box highlights these links. Below the links, there is a list of articles: 1. Read How to Download R and RStudio article, 2. View Install R Packages Video, and 3. Check Hardware Configuration to confirm minimum hardware and memory (Workbench, Connect, Package Manager). A red box highlights these articles. Annotations include: "Keyword Search and Header Links: Every Page" pointing to the search bar, "Click: Site-Map, How-To, Tasks I, Tasks II, Functions, Book, Tidyverse, Debug, Test, Exercises, Data, Pharma, Shiny, Cheat Sheets, Best Practices, Videos, FAQs" pointing to the navigation menu, and "Navigation for Learning: Left to Right and Top to Bottom" pointing to the list of articles.

Keyword Search and Header Links: Every Page

R-Guru Resource Hub

Click: Site-Map, How-To, Tasks I, Tasks II, Functions, Book, Tidyverse, Debug, Test, Exercises, Data, Pharma, Shiny, Cheat Sheets, Best Practices, Videos, FAQs

Home Install & Learn R Common R Tasks Compare with SAS Pharmaverse

Home > Install R Software

What is R and Why Learn R?

Run R Programs to Create Objects

Learn R Programming

Below are s and manag

1. Read How to Download R and RStudio article

2. View Install R Packages Video

3. Check Hardware Configuration to confirm minimum hardware and memory (Workbench, Connect, Package Manager)

Navigation for Learning: Left to Right and Top to Bottom

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

Brand and product names are trademarks of their respective companies.