

# **What is R and Why Should You Learn R?**

**R**





You know that R works great for creating graphs and for performing statistical analysis in Big Data applications. But did you know about the increased utilization of the R programming language for reporting clinical research data?

Have you been sitting on the sidelines and asking yourself what is R and why you should learn it? If so, then you are like thousands of other SAS programmers searching for answers. Learning another programming language might sound like a daunting and time-consuming undertaking. Moreover, learning R is difficult than learning SAS. But like you, I too began by writing my first line of R code. Now I gained a considerable amount of fluency in a language that was once new to me at one time, and I would like to share what I have learned with you now.



# Outline

- 1 **R Programming Webinars**
- 2 **What is R?**
- 3 **What is R Not?**
- 4 **Why Should You Learn R?**
- 5 **How Can You Learn R?**
- 6 **R Interface: R Studio**
- 7 **Compare R with SAS: SASSY Package**
- 8 **Compare R with SQL**
- 9 **R Data Object Process Flow, Structure, Rules and Scope**
- 10 **Common R Packages**
- 11 **R Syntax – Basics**
- 12 **R Examples**
- 13 **Common R Data Frame Operations**
- 14 **Common R FAQs**



# What is R?

R has 'data' centric flexibility. It means that it is built for data cleaning, data management, data analysis and data reporting. It is an ideal language for data scientists and statisticians to have in their toolboxes. Many university students are now required to learn R in their analytical classes.

R processes interchangeable objects. One of the difficulties in learning R is that objects can be abstract with many moving parts. R can be used for quick results with graphs. R first gained traction in graphs since SAS graphs can be hard to customize. R has hundreds of built-in 'intuitively laid out'- based functions such as `randomNames()` to create random names.

R is a popular open-source language with a community of active developers. It means that you can expect to see frequent new R packages and functions released online often. But that means that you will have to be selective about what R packages and functions you choose to use so that you master a unique set of R tasks. Another unique feature of R is that it is a symbol-based programming language. In R, you can directly reference data frame variables using the '\$' symbol and directly subset records using the '[' symbol. It is a game-changer using symbols since referencing variables and creating a subset of records are common programming procedures.

According to the TIOBE Index, R is now ranked 13th amongst the most used of all the programming languages. R is an open-source, object-oriented statistical computing and graphics programming language and environment that allows data analysis, manipulation, graphical reporting in an easy to use and effective way. Because it is open-source, there is massive community support from experienced developers and statisticians that actively contribute towards R's advancement on an ongoing basis. Currently, there is a lot of emphasis on using R within regulated clinical systems as more and more R is becoming an integral part of everyone's toolkit. It is reasonable to assume that going forward R programming skills will be in greater demand within the biometrics field.



## What is R Not?

If you want to learn R, people will let you know that it is not that easy. R is not technically friendly because symbols such as `[]` and syntax are used to perform operations. Syntax is often the most important component to learn. While R got its initial fame from creating simple plots and statistical modelling, it is not restricted only to plots and stats. R is also not just for big data and data science. R can be used for almost any data management and analysis task. R is also not like SAS. R syntax is case sensitive and not friendly towards missing data. It means that you must be careful to submit the correct case as well as inform R how you plan to handle the missing data.

## Why Should You Learn R?

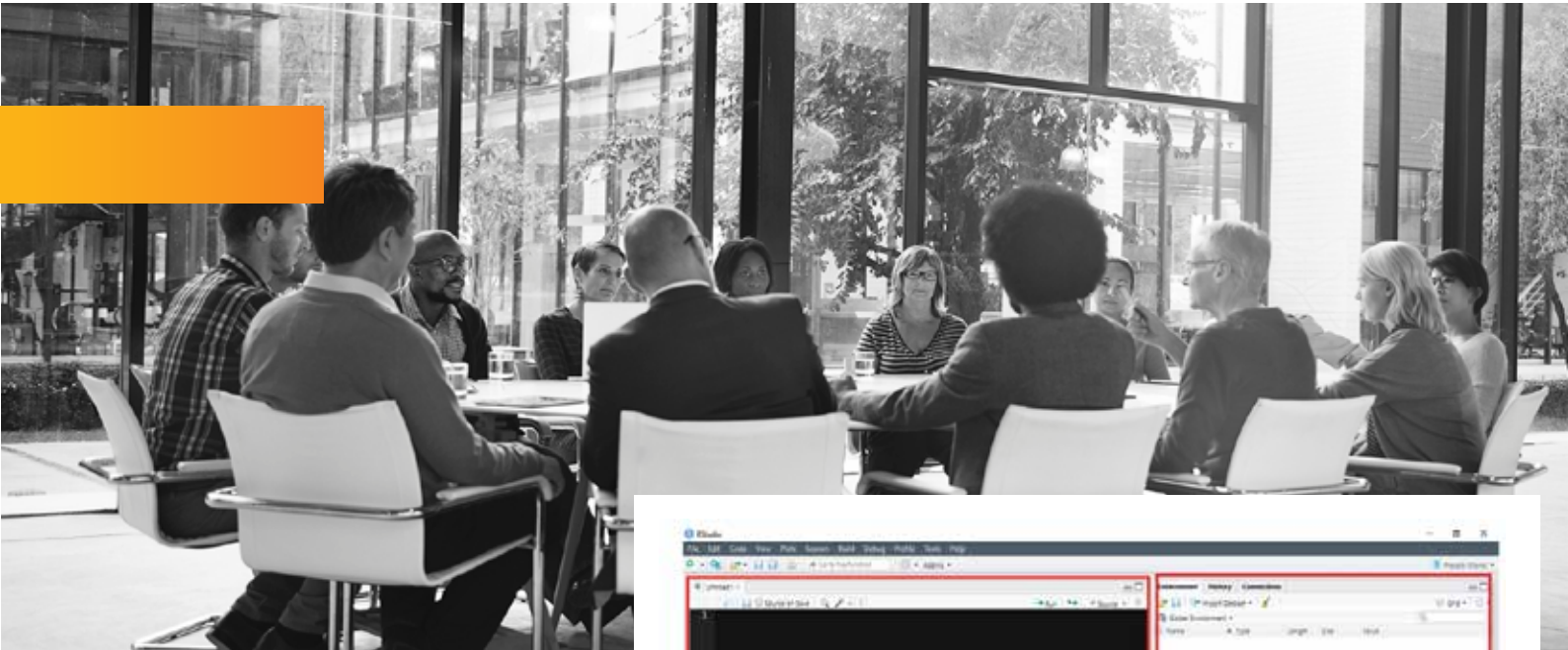
Large pharma companies have already jumped on the bandwagon and now develop their R custom packages. In response to that, CROs now provide internal R training to better support their clients. SAS Institute also recognized the growth of R by integrating some tools with R. CDISC, PhUSE and FDA have also taken steps to install and start using R more. Finally, for SAS programmers wanting to advance in their career, R programming skills is the new requirement for Statistical Programming positions.

## How Can You Learn R?

There are multiple ways to start learning R. The [SASSavvy.com/R](https://SASSavvy.com/R) programming page provides free information about the best R resources. Usually, you will find resources to learn R for all levels. Similarly, you will also find resources to SAS functions, access to R cheat sheets and R Books, as well as access to SAS and R papers. Joining the R programming webinars will help to reinforce your knowledge and understanding of R. The R mentoring programs with one-on-one sessions and guidance are also available on [SASSavvy.com](https://SASSavvy.com).

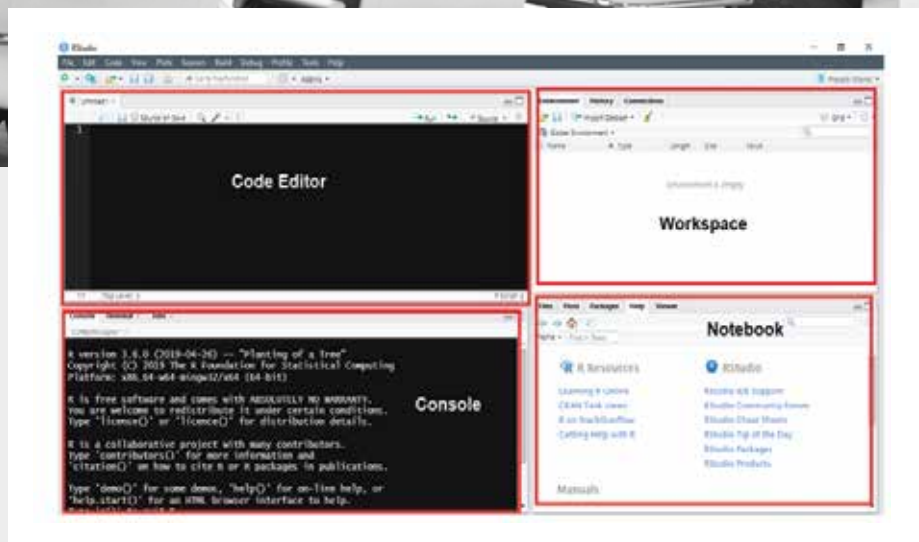
### Below are key points to get you started:

- Install R Today
- Run R Examples
- Understand Basics of R Syntax
- Debug R Programs
- Compare R with SAS
- Compare R with SQL
- Review Pharma Industry Applications



## R Interface: R Studio

The R interface shows four panels: Code Editor, Workspace, Console and Notebook. Each panel has a purpose so that the programmer has full control of his working environment.



**Just like SAS Enterprise Guide, some training is required to start using the R interface.**

### Code Editor window

- Area for code development
- Code will not be evaluated or executed until you hit the 'Run' button

### Workspace window

- Environment / History window
- List objects that exist in the working space
- View comment history (like SAS log)

### Console window

- Code from the script source is evaluated by R
- Allows you to perform quick calculations that you don't need to save

### Notebook window

- Files/Plots/Packages/Help
- Here you can see file folders, plot output, browse and install available packages, access R help



# Compare R with SAS: SASSY Package

With so many SAS programmers learning R, creating an R SAS package was an absolute necessity. With the SASSY package, SAS programmers can have a more seamless transition between R and SAS. With the SASSY package, SAS programmers can almost replicate reviewing logs, datasets, program data steps, formats, and reports. Also, there are R packages and functions to replicate Proc Freq, Proc Means and Proc Report. Though SAS has tools to integrate with R and has packages to replicate SAS programming, the objective of learning R is to consider it a standalone complete toolbox that can be used entirely independent of SAS.



For SAS® programmers, encountering R for the first time can be quite a shock.

- Where is the log?
- Where are my datasets?
- How do I do a data step?
- How do I create a format?
- How do I create a report?

All these basic concepts that were so familiar and easy for you are suddenly gone. How can replacement for SAS®, when it can't even create a decent log!

If you are in this state of shock, or have asked yourself any of the above questions, then the **sassy** system is for you!

## Compare R with SQL

Some of most important R packages include SQL processing. As in SAS, SQL provides powerful query and multitasking functionalities. From my initial review of R's SQL features, I am glad to see a strong correlation between SAS's Proc SQL clauses and R SQL functions. SQL will be a standalone topic covered in a later webinar during this series.

### Example Data

- General Queries
- Aggregate Queries
- Wild card match Queries
- manipulation & Nested Queries
- Join Queries
- Resources



# Common R Packages

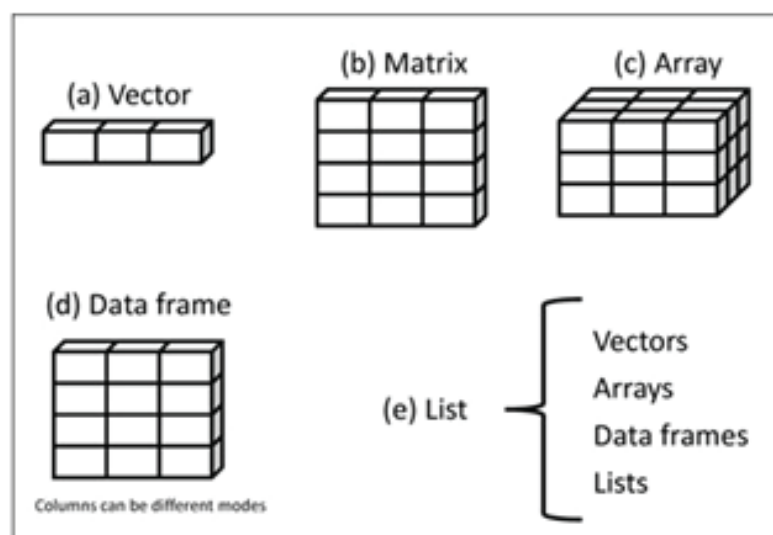
R runs on R packages. R programmers need to identify and run R packages to run R functions. Below is a list of common R packages that perform essential data access, data management and reporting. An overview of R packages is a topic for future webinars.

R Package	Features/Libraries /Functions
base	c(), data.frame()
tidyverse	Common data management package for reshaping, transforming and plotting data tidyr – useful set of functions such as .data[[]] dplyr – useful for data management stringr – string functions purrr – functional programming tibble – modern and effective table system ggplot2 – Popular graphic package readr – read csv files
plyr	ddplyr() – useful to create intermediate steps
sqldf	sql syntax
flextable	Proc Report and ODS type control for report formatting
sassy	logr(), fmtr(), datastep(), reporter(), libr()
officer	Create rtf and powerpoint files
magrittr	Allow %>% for piping operations from one function to another function

## R Syntax – Basics

R is different from SAS in that there are five essential R data structures – vectors, matrix, array, data frames and lists. For SAS programmers, I think the vectors and data frames are the most important. As a basic concept, vectors are like values in a dataset with only one variable, and data frames contain a collection of vectors with many variables and records. Most of all, R processing is performed on data frames. All R data structures, and any outputs created from R functions are R objects.

R data structures





# R Examples

Below are the useful metadata type R functions to describe data frames. Information from R functions displays the data frame variable names, number of records, sample records as well as unique frequency counts and descriptive statistics. The '<=' symbol assigns the tg object the contents of the ToothGrowth data frame. All R functions in the remaining statements process the tg data frame. Any text displayed after the '#' symbol is shown as comments, so ignored by R. These R examples show how R objects are processed by R functions. Simple one function call performs specific tasks. For any of these R functions since the '<=' symbol is not applied, the results are displayed instead of being saved to another R object.

**You can run R statements individually from the Edit > Run Line or Selection option or run all R statements in an R script file from the Edit > Run All option. Results will be displayed in the console window and the tg data frame window will open.**

- `tg <- ToothGrowth` # save sample data frame to tg data frame
- `View(tg)` # browse tg
- `str(tg)` # display tg attributes and sample records
- `attributes(tg)` # display tg attributes, names (tg) is alternative to display variable names
- `head(tg)` # display tg sample records
- `print(tg)` # display tg all records, similar to proc print
- `summary(tg)` # display stats object of continuous variables
- `table(tg)` # display freq of categorical variables

**Below are outputs from the R functions.**

## R Syntax – Data Frame Operations

### View Data Frame

	len	supp	dose
1	4.2	VC	0.5
2	11.5	VC	0.5
3	7.3	VC	0.5
4	5.8	VC	0.5
5	6.4	VC	0.5
6	10.0	VC	0.5
7	11.2	VC	0.5
8	11.2	VC	0.5
9	5.2	VC	0.5
10	7.0	VC	0.5
11	16.5	VC	1.0
12	16.5	VC	1.0
13	15.2	VC	1.0
14	17.3	VC	1.0
15	22.5	VC	1.0
16	17.3	VC	1.0
17	13.6	VC	1.0
18	14.5	VC	1.0
19	18.8	VC	1.0

### R Functions are similar to SAS Procedures

```
> stage <- summary(tg)
> print(stage)

len      supp      dose
Min.   4.20  VC=30   Min.   0.500
1st Qu.13.07  VC=30   1st Qu.0.500
Median 19.28             Median 1.000
Mean   16.01             Mean   1.167
3rd Qu.25.27             3rd Qu.2.000
Max.   33.90             Max.   2.000

> freq <- table(tg)
> print(freq)
      dose = 0.5
len      supp
4.2  0  1
5.2  0  1
5.8  0  1
6.4  0  1
7.0  0  1
7.3  0  1
10.0 0  1
11.2 0  1
11.2 0  1
13.6 0  1
14.5 0  1
15.2 1  0
16.5 1  0
16.5 1  0
17.3 1  0
17.3 1  0
18.8 1  0
22.5 1  0

> attributes(tg)
$names
[1] "len" "supp" "dose"

$dose
[1] "Data.frame"

$row.names
[1] 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[24] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
[50] 51 52 53 54 55 56 57 58 59 60
```

Descriptive Stats

Frequency Counts

Data Frame Variable Names and Rows

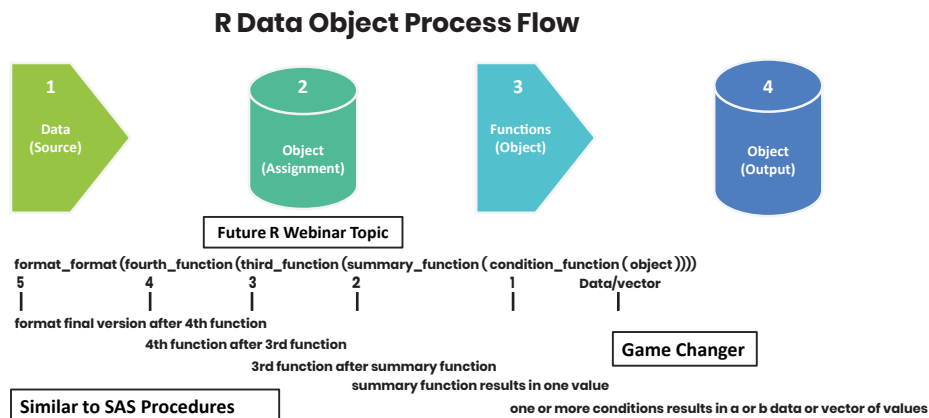
	len	supp	dose
1	4.2	VC	0.5
2	11.5	VC	0.5
3	7.3	VC	0.5
4	5.8	VC	0.5
5	6.4	VC	0.5
6	10.0	VC	0.5

Sample Records

# R Data Object Process Flow, Structure, Rules and Scope

R processes are a collection of R functions that create and read R objects until the final object is created. The output of the first R function is the input to the second R function.

The output of the second R function is input to the third R function. In the diagram below, this continues to the fifth outer R function. This means that R functions are used to access, manage, transform, and then summarize data. In addition, because of this unique software architecture, R functions can have multiple nested levels.



R programming is unique since R objects store data which must be valid data types so that R functions can be applied to create new R objects. So, in a sense, given the software's unique architecture, the prior level of data within an object must be fully validated to advance to the next level of data contained within the subsequent object. With symbols, direct references can be made to variables as independent objects. This makes R programming more flexible. Along with R base, tidyverse, and ggplots are popular R packages. R data object process flow and structure, rules and scope are topics for future webinars.

## R Structure, Rules and Scope





## Common R Data Frame Operations

As mentioned earlier, R leverages symbols to perform common tasks. Below is a list of common R data frame operations.

Data Frame Operations/Functions	Symbol / Syntax
Create Data in Objects	<code>c()</code>
Data Frame Variable Reference	<code>\$</code>
Index Subset (Variable or Records)	<code>[]</code>
Format	<code>format()</code> to format variables and data types <code>sprintf()</code> to format reporting columns <code>str_glue()</code> to format char, num and dates as string
Combine Data Frames by Rows	<code>rbind()</code> to append data frames
Combine Data Frames by Columns	<code>cbind()</code> to merge vectors
Merge Data Frames	<code>merge()</code> , <code>left_join()</code> , <code>sqldf()</code>
SQL Clauses	<code>select()</code> , <code>mutate()</code> & <code>chng=</code> , <code>ifelse()</code> , <code>filter()</code> , <code>group_by()</code> , <code>arrange()</code>
SQL Operations	<code>ddplyr()</code> , <code>dplyr</code> , <code>sqldf()</code>
Transpose Data	<code>pivot_wider()</code> – tidyverse <code>pivot_longer()</code> – tidyverse Old – <code>melt()</code> , <code>spread()</code>
Data Type Conversion	<code>as.character()</code> , <code>as.date()</code> , <code>as.integer()</code> , <code>as.data.frame()</code> , <code>as.logical()</code> , <code>as.table()</code> , <code>as.array()</code>
Missing Values	<code>is.na()</code> , <code>is.null()</code>
Object Metadata	<code>is.list()</code> , <code>is.matrix()</code> , <code>is.vector()</code> , <code>is.data.frame()</code> , <code>is.array()</code>
Characterize Data / Variable Metadata	<code>View()</code> , <code>attributes()</code> , <code>length()</code> , <code>ncol()</code> , <code>nrow()</code> , <code>names()</code> , <code>str()</code> , <code>typeof()</code> , <code>class()</code> , <code>head()</code> , <code>print()</code> , <code>summary()</code> , <code>table()</code>
View Data Frame	<code>View()</code> – upper case 'V', base R <code>view()</code> – lower case 'v', tidyverse
Loop through List or Variable	<code>for (i in &lt;data_frame&gt;\$&lt;variable_name&gt;)</code>
Custom Functions	<code>function((condition) {true} else {false})</code>

# Common R FAQs

While learning R, all programmers have common FAQs. Below is a brief list of high-level common R FAQs. See [SASSavvy.com/common\\_faq\\_index.html](https://SASSavvy.com/common_faq_index.html) for a comprehensive list. SASSavvy.com welcomes your R questions.

- › Do I have to learn all R packages?
- › Are Statistical Programming positions requiring R Skills?
- › Can R read SAS datasets?
- › Is R better than SAS for creating graphs?
- › Can you use R to create SDTMs, ADaMs?
- › Can you use R to create publication quality summary and listings?
- › Can you use R for validation?
- › Does FDA use R?
- › Is R validated?



## Summary

There is an alternative way to write statistical programs, i.e., SAS is no longer the only software used for creating SDTMs, ADaMs and TLGs. R has evolved from the early days of quick plots to a more powerful and mainstream programming language that has been accepted by the SAS programming community. As more and more pharmaceutical companies and CROs begin to recognize this new trend, the more they will likely prepare their teams for new and innovative ways of analyzing clinical research data.

## Author




Sunil Gupta, MS, is an international speaker, best-selling author of 5 SAS books, and a global SAS and CDISC corporate trainer. Sunil has over twenty-five years of experience in the pharmaceutical industry. Recently, Sunil has been teaching a CDISC online class at the University of California, San Diego. He teaches Data Science using SAS at UCLA and UCSD Extension.

In 2019, Sunil published his fifth book titled 'Clinical Data Quality Checks for CDISC Compliance Using SAS' and in 2011, Sunil launched his unique SAS mentoring blog, [SASSavvy.com](https://SASSavvy.com), for smarter SAS searches. Sunil holds an MS in Bioengineering from Clemson University and a BS in Applied Mathematics from the College of Charleston.

**Sponsored by ACL Digital Life Sciences.**

ACL Digital is a design-led Digital Experience, Product Innovation, Engineering and Enterprise IT offerings leader. From strategy, to design, implementation and management we help accelerate innovation and transform businesses. ACL Digital is a part of ALTEN group, a leader in technology consulting and engineering services.

[business@acldigital.com](mailto:business@acldigital.com) | [www.acldigital.com](https://www.acldigital.com)

USA | UK | France | India   

Proprietary content. No content of this document can be reproduced without the prior written agreement of ACL Digital. All other company and product names may be trademarks of the respective companies with which they are associated.

