What is R and Why Should You Learn R?
Sunil Gupta, Founder of R-Guru.com and SASSavvy.com

You probably already know that R works great for creating graphs and for performing statistical analyses 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 dauting and time-consuming undertaking. Moreover, learning R is more 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've learned with you now.

The upcoming webinar series 'R Programming for Biometrics Professionals' is a logically prepared step-by-step crash course that will immerse you in the world of R programming. This upcoming series will be sponsored by ACL Digital Life Science.

Outline

- R Programming Webinars
- What is R?
- What is R Not?
- Why Should You Learn R?
- How Can You Learn R?
- R Interface: R Studio
- Compare R with SAS: SASSY Package
- Compare R with SQL
- R Data Object Process Flow, Structure, Rules and Scope
- Common R Packages
- R Syntax – Basics
- R Examples
- Common R Data Frame Operations
- Common R FAQs

What is R?

R has 'data' centric flexibility. This 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 very popular open-source language with a community of active developers. This means that you can expect to see frequent new R packages and functions released on-line often. But that means that you are going to 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 is that R 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. This is a game changer using symbols since referencing variables and creating 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's open source there is large 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?

Anyone starting to learn R will let you know that R is not easy to learn. R is not technically friendly because symbols such as [] along with syntax are used to perform operations. In fact, the syntax is often the most important component to learn. While R got its initial fame from creating simple plots and statistical modeling, R is not restricted to plots and stats. R is also not just for big data and data science. R can be used for most any data management and analysis tasks. R is also not similar to SAS. R syntax is case sensitive and not friendly towards missing data. This means that you must be careful to submit the correct case as well as inform R how you plan to handle missing data.

Why Should You Learn R?

Large pharma companies have already jumped on the bandwagon and now develop their own 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 are starting to be a new requirement for Statistical Programming positions.

How Can You Learn R?

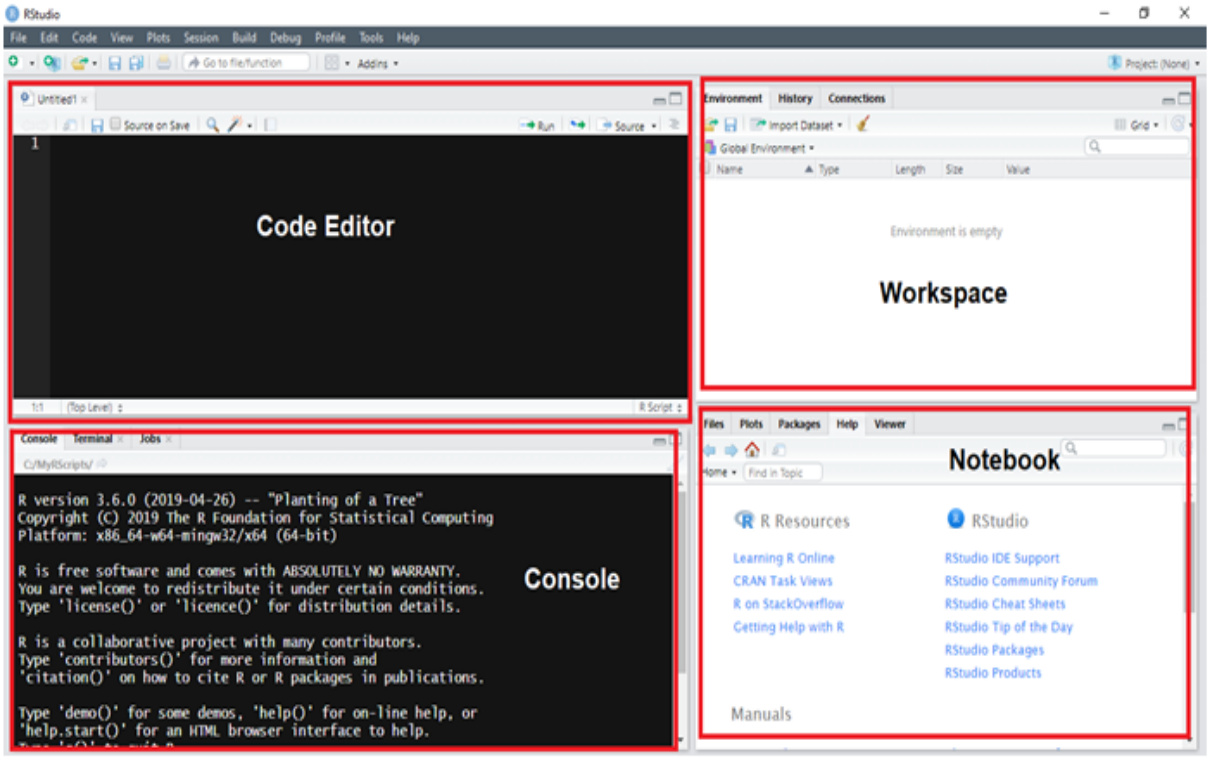
There are several steps you can take now to start learning R. The SASSavvy.com/R_programming page provides free information about best R resources. You can find great resource to learn R for all levels, similar SAS functions, access to R cheat sheets and R Books as well as access SAS and R papers. Joining the R programming webinars will help to reinforce your knowledge and understanding of R. R mentoring programs with one-on-one sessions and guidance are also available on 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

- Ares 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

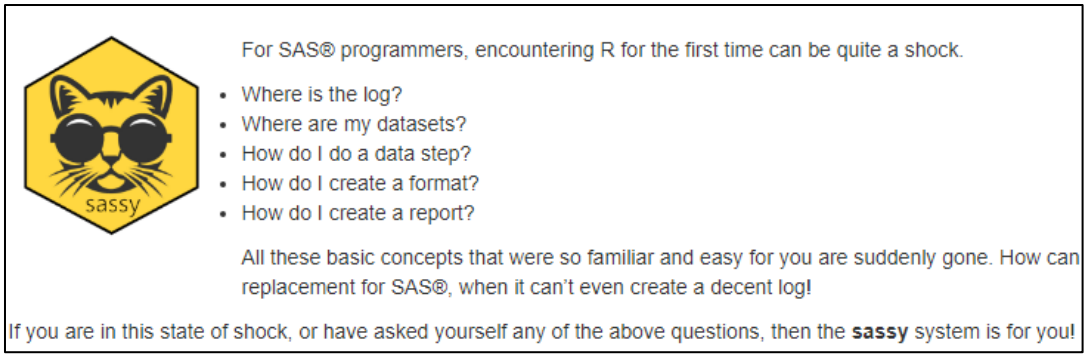
- Here code from the script source is evaluated by R
- Also allow you to perform quick calculations that you don't need to save

Notebook window

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

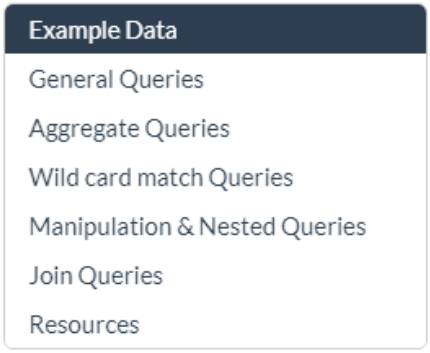
Compare R with SAS: SASSY Package

With so many SAS programmers learning R, it makes sense that an R SAS package was created. 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, reviewing datasets, and program data steps, formats and reports. There are R packages and functions to replicate Proc Freq, Proc Means and Proc Report. It is important to realize that although SAS has tools to integrate with R and R has packages to replicate SAS programming, the objective of learning R is to consider it a standalone complete toolbox in its own right that can be used entirely independent of SAS.



Compare R with SQL

Some of most important R packages include SQL processing. As in SAS, SQL provides a very 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.



Common R Packages

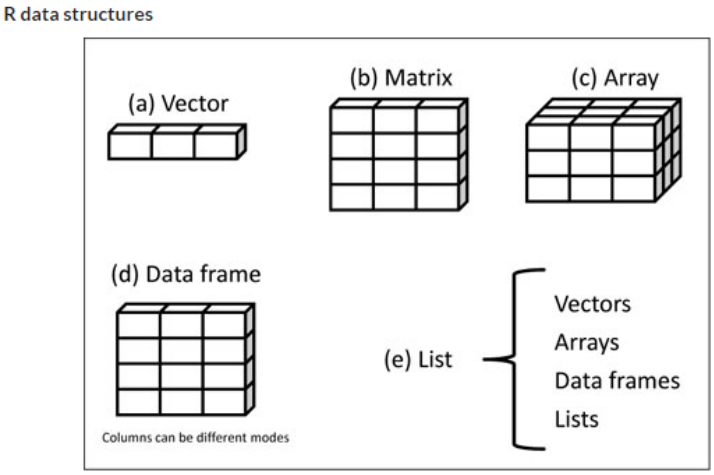
R runs on R packages. R programmers need to identify and run R packages to run R functions. Below is a list of common R packages that perform essential data access, data management and reporting.
An overview of R packages is a topic for future webinars.

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 purr – 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
magtrittr	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 similar to values in a dataset with only one variable and data frames contain a collection of vectors so contain many variables and records. Most all R processing is performed on data frames. All R data structures and any outputs created from R functions are R objects.



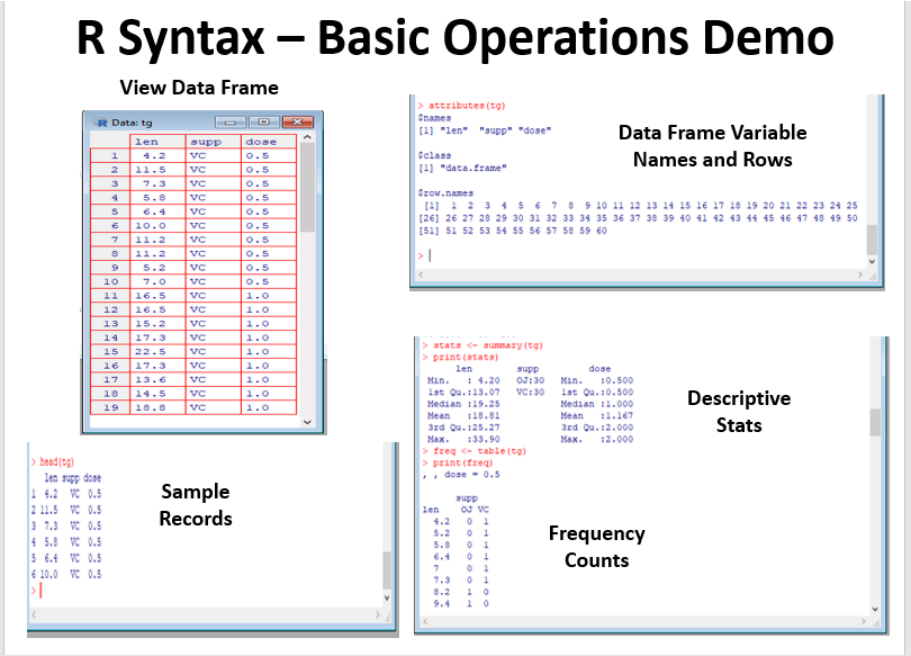
R Examples

Below are useful metadata type R functions to describe data frames. Information from R functions display 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 after the '#' symbol are 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.

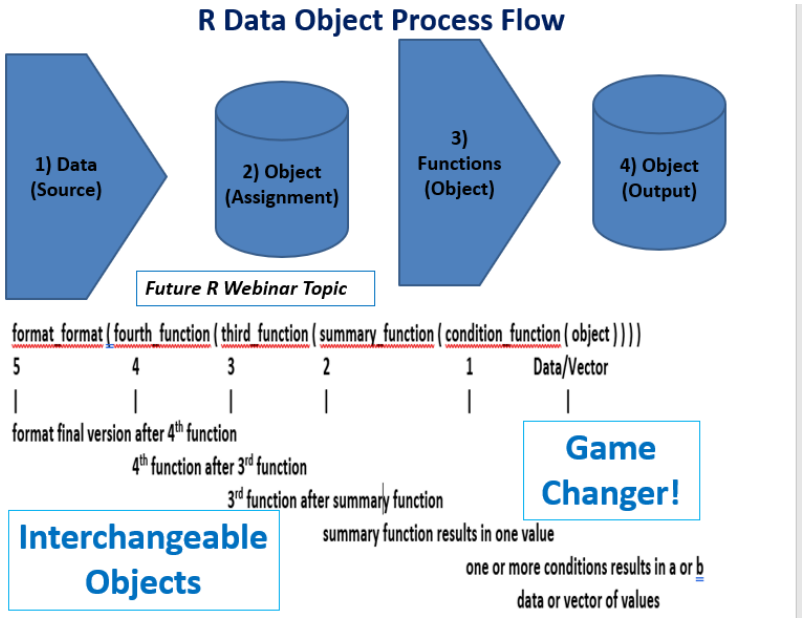
> **tg** <- ToothGrowth # save sample data frame to tg data frame
> **View**(tg) # browse tg
> **str**(tg) # display tg attributes and sample records
> **attributes**(tg) # display tg attributes, names (tg) is alternative to display variable names
> **head**(tg) # display tg sample records
> **print**(tg) # 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.



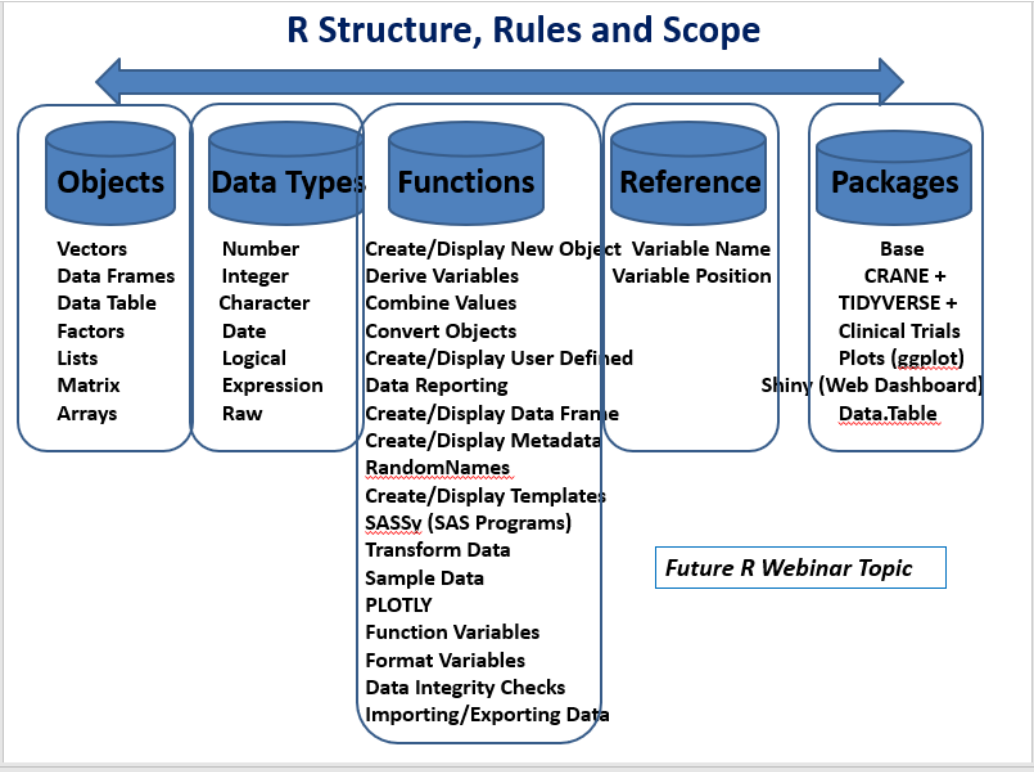
R Data Object Process Flow, Structure, Rules and Scope

R processes are basically 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. 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 in order 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.



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	c()
Data Frame Variable Reference	$
Index Subset (Variable or Records)	[]
Format	format() to format variables and data types sprintf() to format reporting columns str_glue() to format char, num and dates as string
Combine Data Frames by Rows	rbind() to append data frames
Combine Data Frames by Columns	cbind() to merge vectors
Merge Data Frames	merge(), left_join(), sqldf()
SQL Clauses	select(), mutate() & chg=, ifelse(), filter(), group_by(), arrange()
SQL Operations	ddplyr(), dplyr, sqldf()
Transpose Data	pivot_wider() – tidyverse pivot_longer() - tidyverse Old - melt(), spread()
Data Type Conversion	as.character(), as.date(), as.integer(), as.data.frame(), as.logical(), as.table(), as.array()
Missing Values	is.na(), is.null()
Object Metadata	is.list(), is.matrix(), is.vector(), is.data.frame(), is.array()
Characterize Data / Variable Metadata	View(), attributes(), length(), ncol(), nrow(), names(), str(), typeof(), class(), head(), print(), summary(), table()
View Data Frame	View() – upper case 'V', base R

	view() – lower case ‘v’, tidyverse
Loop through List or Variable	for (i in <data_frame>$<variable_name>)
Custom Functions	function((condition) {true} else {false})

Common R FAQs

While learning R, all programmers have common FAQs. Below is a brief list high level common R FAQs. See 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

Now more than ever, 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.

Biography

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. Most recently, Sunil is teaching a CDISC online class at the University of California at San Diego and classes on Data Science using SAS at UCLA and UCSD Extension. In 2019, Sunil published his fifth book, Clinical Data Quality Checks for CDISC Compliance Using SAS and in 2011, Sunil launched his unique SAS mentoring blog, SASSavvy.com, for smarter SAS searches. Sunil has MS in Bioengineering from Clemson University and a BS in Applied Mathematics from the College of Charleston.