R Made Easier for SAS Programmers

Sunil Gupta, Founder of R-Guru.com,
Author and CDISC SME
GuptaProgramming@gmail.com

What Does R Mean to You?

Open Source

Type of Tasks

Free Software

Learning Curve

Validation

Pharmaverse

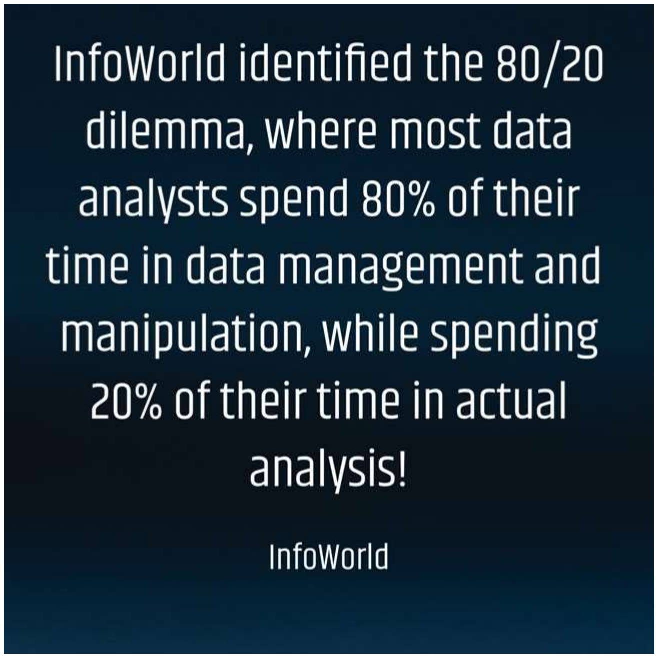
R Shiny Reviews

FDA

Programming

Hundreds of R Packages

Replace SAS or Enhance w/ R?



InfoWorld identified the 80/20 dilemma, where most data analysts spend 80% of their time in data management and manipulation, while spending 20% of their time in actual analysis!

InfoWorld

While Learning R, Avoid these Objects

❌ Topic: Data Science Applications
- Database Programming is more direct and more focused

❌ Variable Types: Integers, Lists, Loops, Logical
- Use Numeric, Character and Date Variable Types

❌ Objects: Vectors, Arrays, Matrix, row_names()
- Use Data Frames and observation numbers

❌ Packages: Non-common, similar multiple
- Best to pick one package/function for each type of task
- Tidyverse and dply functions instead of base R functions
- Do not re-invent the wheel, search for packages
- QC each package before use in production

❌ Functions: same name, attach(), loops, user defined R functions
- Use common R packages, functions and defaults

Upside Down World

- ✓ 'Data' Centric Flexibility

  Data Cleaning, Data Management, Data Analysis, Data Reporting, Graphs

  Ideal for Data Science

  Statistician's Toolbox

What is R?

Similar to SAS Procedures / Excel Functions

No Variable Lengths in R

(Requires valid parameters, inputs and syntax to get results)

- ✓ Process Interchangeable Objects

  Abstract, Many Moving Parts

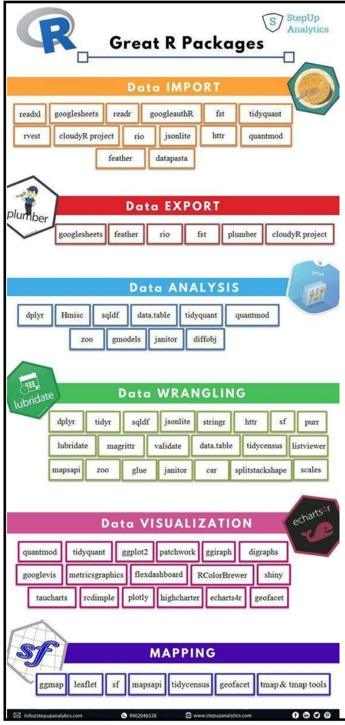
- ✓ 100's of Built-in 'Intuitive'-Based Functions

  NA represent missing values

- ✓ Symbol-Based Programming Language

  $ references variables

  [] is used for subsetting



Great R Packages

StepUp Analytics

Data IMPORT

readxl, googlesheets, readr, googleauthR, fst, tidyquant, rvest, cloudyR project, rio, jsonlite, httr, quantmod, feather, datapasta

Data EXPORT

googlesheets, feather, rio, fst, plumber, cloudyR project

Data ANALYSIS

dplyr, Hmisc, sqldf, data.table, tidyquant, quantmod, zoo, gmodels, janitor, diffobj

Data WRANGLING

dplyr, tidyr, sqldf, jsonlite, stringr, httr, sf, purr, lubridate, magrittr, validate, data.table, tidycensus, listviewer, mapsapi, zoo, glue, janitor, car, splitstackshape, scales

Data VISUALIZATION

quantmod, tidyquant, ggplot2, patchwork, ggiraph, digraphs, googlevis, metricsgraphics, flexdashboard, RColorBrewer, shiny, taucharts, rcdimple, plotly, highcharter, echarts4r, geofacet

MAPPING

ggmap, leaflet, sf, mapsapi, tidycensus, geofacet, tmap & tmap tools

Info@stepupanalytics.com 9902046528 www.stepupanalytics.com

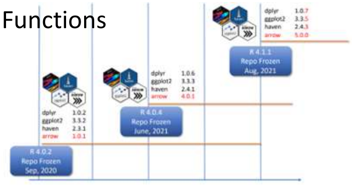
Welcome to the World of R Packages

R Community Journey vs SAS Silos

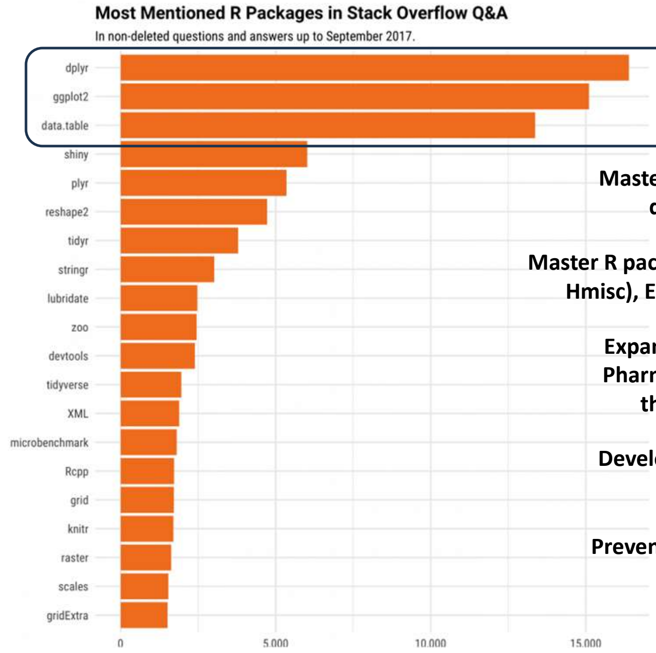
- Common Goals (Destination)
- Common Challenges (Reality)
- R Developer (Driver, Visionary)
- R Programmers (Passengers, Fans)
- Validation (# Downloads, Usage)
- GitHub (Central Version Control)
- Documentation (Vignette Pages)

History and Challenges of R Packages

- End-to-End Data Processing is Possible
- Leverage Common Packages, Functions & Defaults
- Pharmaverse Packages, Metadata & Tools
- R Function Conflicts & Dependencies
- Invalid R Function Parameters
- Can not find R Functions



Popular & Common R Packages



80% of R programmers use 20% of R packages

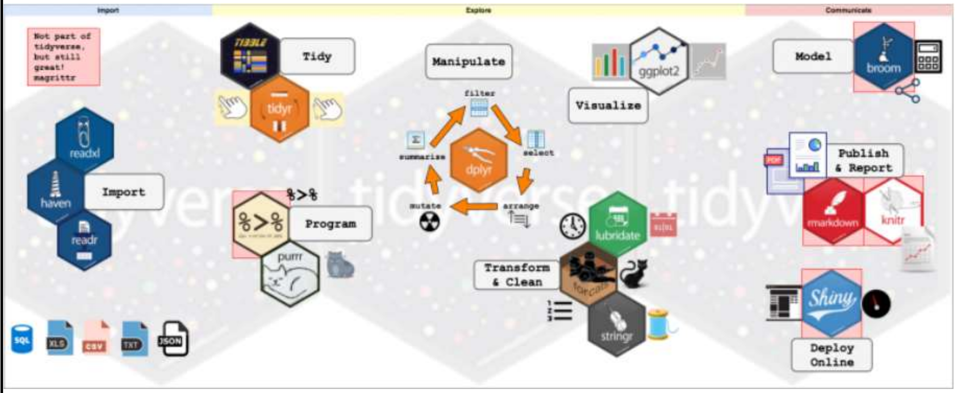
Master Base R, Tidyverse, dplyr & ggplot2

Master R packages for SAS (haven, Hmisc), Excel and CSV Files

Expand to Learn & Leverage Pharmaverse R packages for the pharma industry

Develop in-house R packages or functions

Prevention is the best Defense



Tidyverse: There is an R Package for That!

<u>Community of Developers</u>

- Import
- Data Clean, Data Manipulation
- Program Language
- Visualize, Statistical Models
- Publish, Web Applications

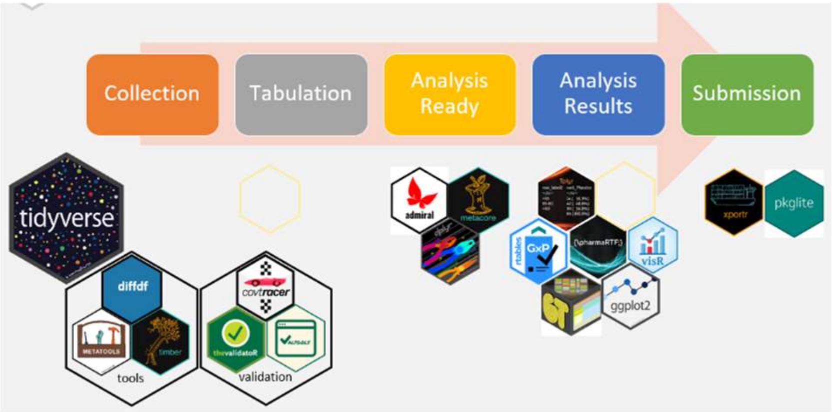
Clinical Trial Process Workflow:
There is a pharmaverse Package for that!

Organizations can continue to use in-house R packages & functions!

Tidyverse

Pharmaverse

Smarter organizations can select packages to fill gaps!



9

Validation of R Packages/Functions/SAS

Risk Assessment: Identify & Minimize

IQ/OC QC Test Plan

- One Data Input
- One Specification
- Run R1, R2 and SAS
- Compare the Accuracy of Expected Results
- No Errors, Warnings

Factors

- Task: Input, Management, etc.
- Stable R Package Version
- R Package Dependency
- # of Downloads
- R Package/Function Usage
- Examples and User Documentation

Accuracy – *How close do R and SAS numbers match?*

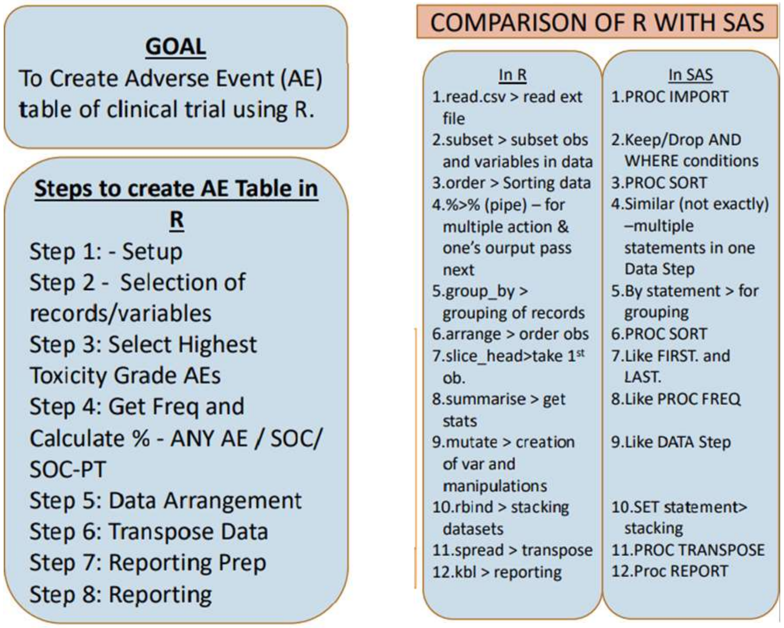
Traceability – *Can you document and audit the process to match the specifications?*

Reproducibility – *Is their control in R packages, versions, functions and parameters?*

Ready to Leverage *R Advantages* over *SAS*?

- **Direct Replacement of SAS**
  =
  - Leadership commitment to R
  - data input (excel, csv, etc.), management, analysis, reporting (pdf, excel, csv, rtf, etc.) and graphing (Tidyverse, DPLYR, magrittr (%>% Piping), Haven, Hmisc, readxl, writexl, GT & ggplot2)
- **Remove Pain Points in SAS**
  −
  - Graphs vs SAS's Graphical Template Language (GTL)
- **Enhance Work Flow Process**
  +
  - Data Frame options, Derivations based on Direct Variable and Row reference
  - Pharmaverse R Packages vs SAS in-house silos
  - R Shiny Apps vs SAS's Dashboards
  - Latest Statistical Modeling R packages
  - Easily create dummy data

R and SAS have similar Work Flow Process



GOAL

To Create Adverse Event (AE) table of clinical trial using R.

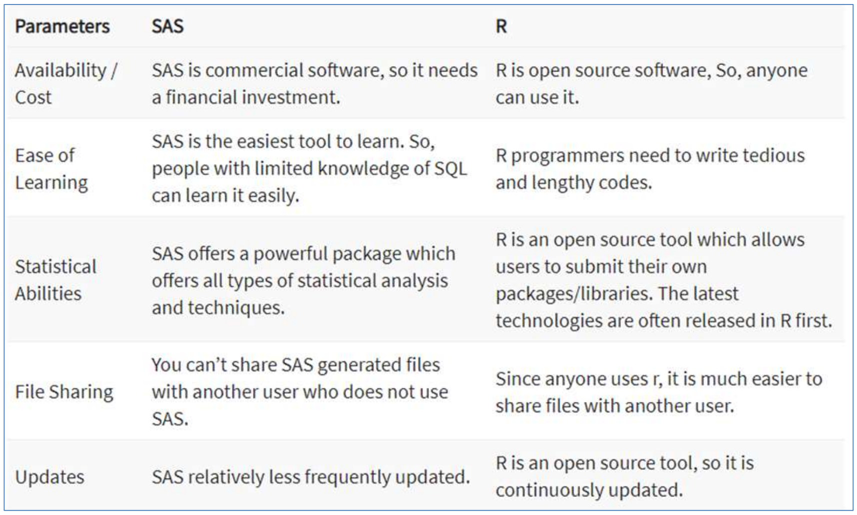
Steps to create AE Table in R

Step 1: - Setup
Step 2 - Selection of records/variables
Step 3: Select Highest Toxicity Grade AEs
Step 4: Get Freq and Calculate % - ANY AE / SOC/ SOC-PT
Step 5: Data Arrangement
Step 6: Transpose Data
Step 7: Reporting Prep
Step 8: Reporting

COMPARISON OF R WITH SAS

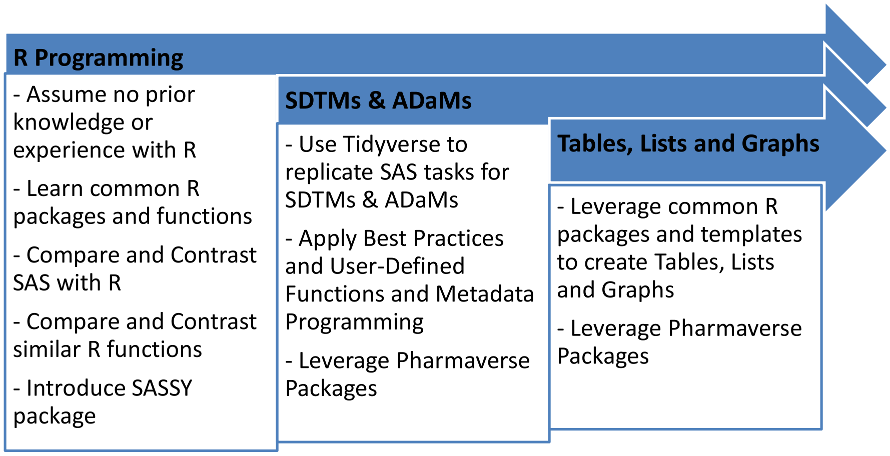
In R	In SAS
1.read.csv > read ext file	1.PROC IMPORT
2.subset > subset obs and variables in data	2.Keep/Drop AND WHERE conditions
3.order > Sorting data	3.PROC SORT
4.%>% (pipe) – for multiple action & one's ourput pass next	4.Similar (not exactly) –multiple statements in one Data Step
5.group_by > grouping of records	5.By statement > for grouping
6.arrange > order obs	6.PROC SORT
7.slice_head>take 1st ob.	7.Like FIRST. and LAST.
8.summarise > get stats	8.Like PROC FREQ
9.mutate > creation of var and manipulations	9.Like DATA Step
10.rbind > stacking datasets	10.SET statement> stacking
11.spread > transpose	11.PROC TRANSPOSE
12.kbl > reporting	12.Proc REPORT

Fully Understand Key Differences between R and SAS



Parameters	SAS	R
Availability / Cost	SAS is commercial software, so it needs a financial investment.	R is open source software, So, anyone can use it.
Ease of Learning	SAS is the easiest tool to learn. So, people with limited knowledge of SQL can learn it easily.	R programmers need to write tedious and lengthy codes.
Statistical Abilities	SAS offers a powerful package which offers all types of statistical analysis and techniques.	R is an open source tool which allows users to submit their own packages/libraries. The latest technologies are often released in R first.
File Sharing	You can't share SAS generated files with another user who does not use SAS.	Since anyone uses r, it is much easier to share files with another user.
Updates	SAS relatively less frequently updated.	R is an open source tool, so it is continuously updated.

R-Guru Learning Process for SAS Programmers in the Pharma and Medical Device Industries
R (Tidyverse, DPLYR) > SDTMs & ADaMs > TLGs



R Programming

- Assume no prior knowledge or experience with R
- Learn common R packages and functions
- Compare and Contrast SAS with R
- Compare and Contrast similar R functions
- Introduce SASSY package

SDTMs & ADaMs

- Use Tidyverse to replicate SAS tasks for SDTMs & ADaMs
- Apply Best Practices and User-Defined Functions and Metadata Programming
- Leverage Pharmaverse Packages

Tables, Lists and Graphs

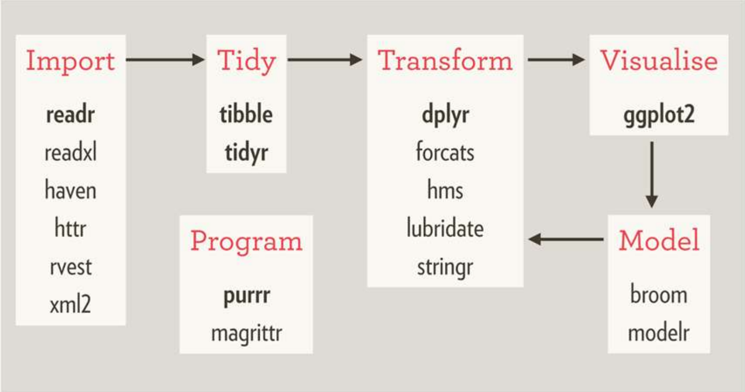
- Leverage common R packages and templates to create Tables, Lists and Graphs
- Leverage Pharmaverse Packages

Tidyverse has Most all Relevant R Libraries!

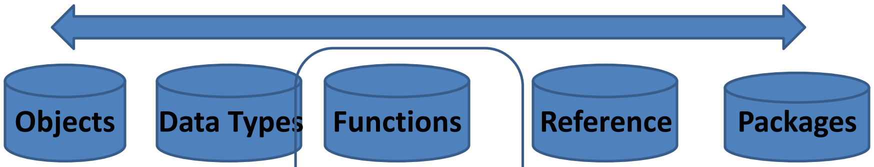
One R Package has it All!

Working Folder: setwd("C:/study/analysis/output") # create working folder in advance
getwd() # confirm new working folder, default C:/Users/Sunil/Documents
# important for using saveRDS and readRDS functions for storing & reading files

Write Function: write.csv(dm, "C:/study/analysis/output/mydm.csv", row.names = FALSE)
Save Function: saveRDS(asl2, file = "asl2.RDS") # save asl2 as permanent data frame
Read Function: myasl <- readRDS("asl2.RDS") # read permanent asl2 data frame



R Structure, Rules and Scope

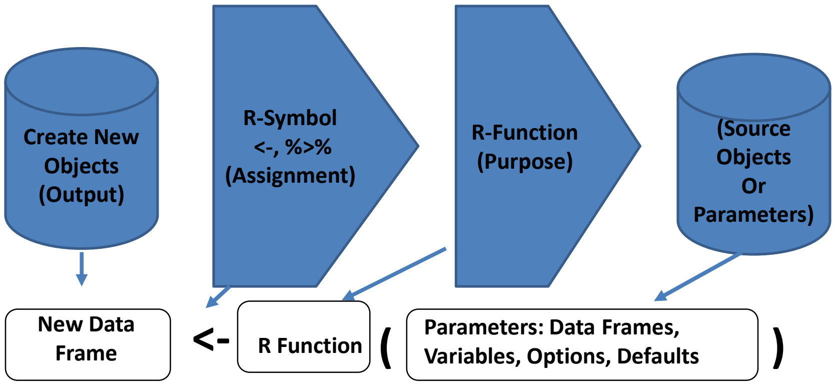


Objects	Data Types	Functions	Reference	Packages
Vectors	Number	Create/Display New Object	Variable Name	Base
Data Frames	Integer	Derive Variables	Variable Position	CRANE +
Data Table	Character	Combine Values		TIDYVERSE +
Factors	Date	Convert Objects		Clinical Trials
Lists	Logical	Create/Display User Defined		Plots (ggplot)
Matrix	Expression	Data Reporting		Shiny (Web Dashboard)
Arrays	Raw	Create/Display Data Frame		Data.Table
		Create/Display Metadata		
		RandomNames		
		Create/Display Templates		
		SASSy (SAS Programs)		
		Transform Data		
		Sample Data		
		PLOTLY		
		Function Variables		
		Format Variables		
		Data Integrity Checks		
		Importing/Exporting Data		

R is a ‘Syntax and Function’ based Language

R Data Object Process Flow: Validated R Packages

Assure open and closed brackets: [], (), ‘’. Close bracket defines end of R-command. %>% saves time from creating intermediate objects.



Create New Objects (Output)

R-Symbol <-, %>% (Assignment)

R-Function (Purpose)

(Source Objects Or Parameters)

New Data Frame <- R Function (Parameters: Data Frames, Variables, Options, Defaults)

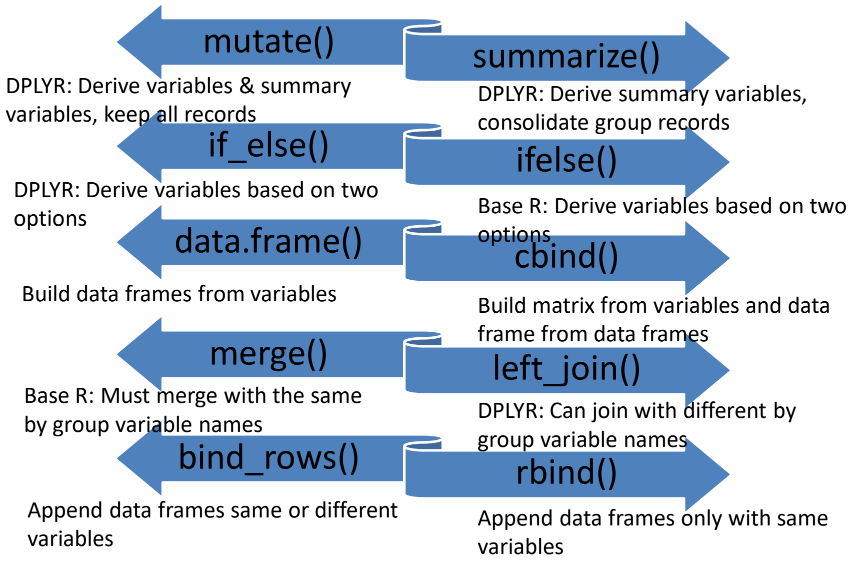
Requires Valid Object Names, Symbols, Functions, Parameters and Objects

One ‘Function’ Away

```
my_data <- cbind(usubjid, age, date, in_study)
New_R_Object <- R_Function(R_Objects)

R_Function(Parameter 1, Parameter 2, etc.)
R Keyword(Existing R_Objects)
```

R Functions: Compare and Contrast Key Differences



mutate()
DPLYR: Derive variables & summary variables, keep all records

summarize()
DPLYR: Derive summary variables, consolidate group records

if_else()
DPLYR: Derive variables based on two options

ifelse()
Base R: Derive variables based on two options

data.frame()
Build data frames from variables

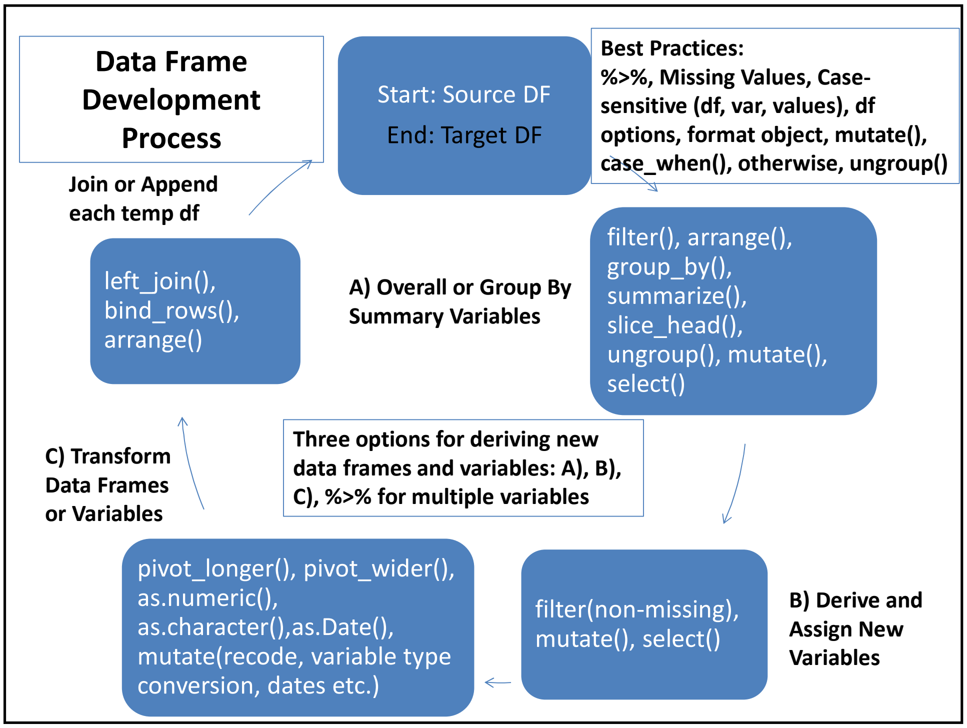
cbind()
Build matrix from variables and data frame from data frames

merge()
Base R: Must merge with the same by group variable names

left_join()
DPLYR: Can join with different by group variable names

bind_rows()
Append data frames same or different variables

rbind()
Append data frames only with same variables



Data Frame Development Process

Start: Source DF

End: Target DF

Best Practices:
%>%, Missing Values, Case-sensitive (df, var, values), df options, format object, mutate(), case_when(), otherwise, ungroup()

A) Overall or Group By Summary Variables

filter(), arrange(), group_by(), summarize(), slice_head(), ungroup(), mutate(), select()

B) Derive and Assign New Variables

filter(non-missing), mutate(), select()

C) Transform Data Frames or Variables

pivot_longer(), pivot_wider(), as.numeric(), as.character(),as.Date(), mutate(recode, variable type conversion, dates etc.)

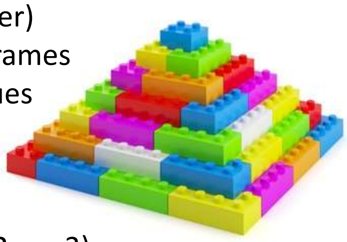
Join or Append each temp df

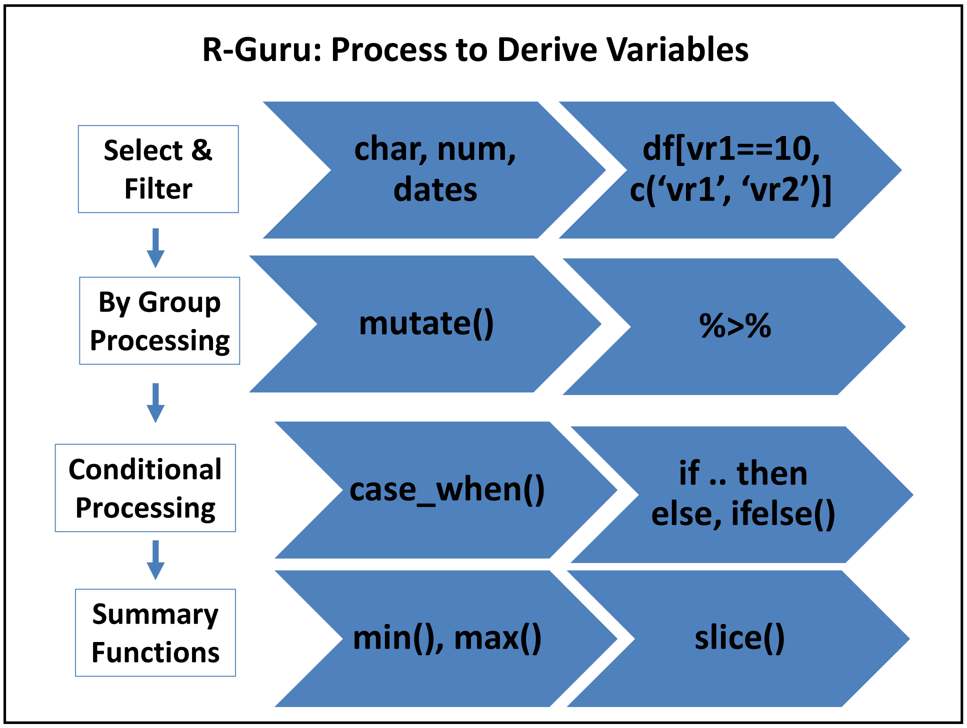
left_join(), bind_rows(), arrange()

Three options for deriving new data frames and variables: A), B), C), %>% for multiple variables

Build Objects to Create R Deliverables Checklist

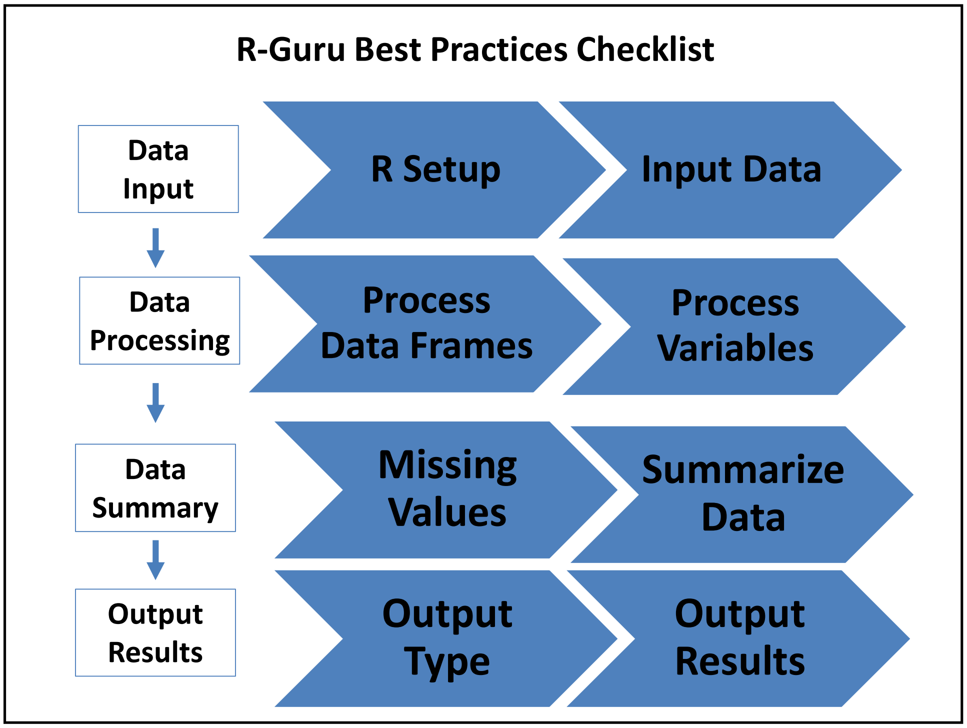
- ❑ Stream Functions with %>%
- ❑ Apply Data Frame Options (Select & Filter)
- ❑ Query (Select & Filter) and View Data Frames
- ❑ Assign Variable Names and Missing Values
- ❑ Derive Variables Based on Conditions
  - ❑ Based on Constants with mutate()
  - ❑ Based on Variables
    - ❑ Var4 = Ifelse(var1 condition, var2, var3)
    - ❑ Left Variable Assignments with df$vr and nested ifelse()
  - ❑ Summary Variables
    - ❑ Add Variables with mutate()
    - ❑ Consolidate Records with summarize(), slice()
    - ❑ Overall or By Group Variables with summary functions
- ❑ Join Objects with left_join()
- ❑ Append Objects (Even, Uneven)
- ❑ Transpose Variables (Rows, Columns)





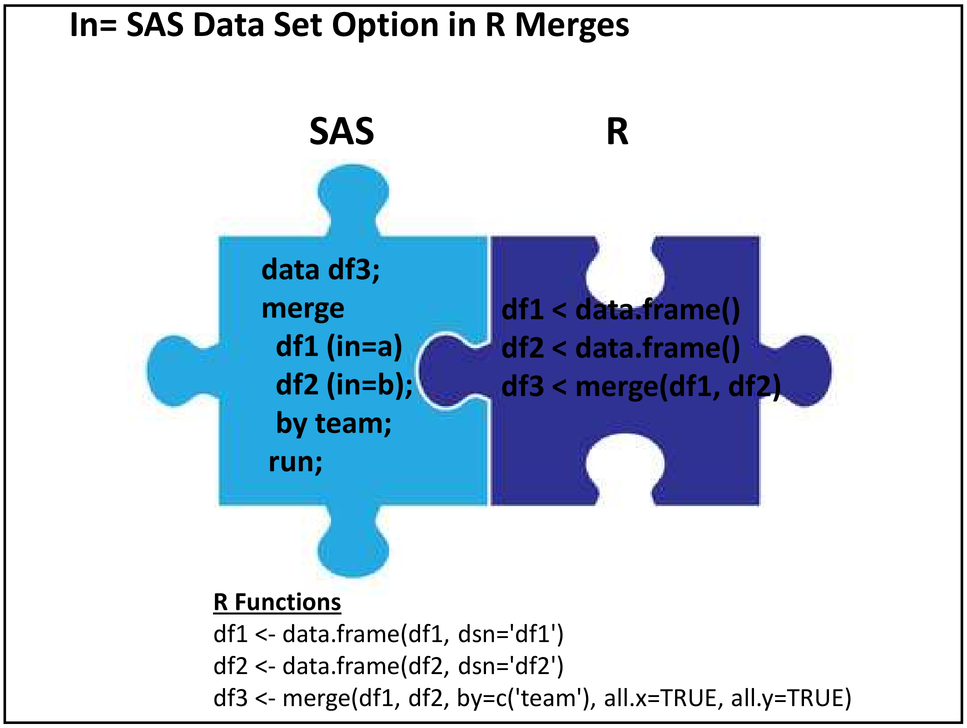
R-Guru: Process to Derive Variables

Select & Filter	char, num, dates	df[vr1==10, c('vr1', 'vr2')]
By Group Processing	mutate()	%>%
Conditional Processing	case_when()	if .. then else, ifelse()
Summary Functions	min(), max()	slice()



R-Guru Best Practices Checklist

Data Input	R Setup	Input Data
Data Processing	Process Data Frames	Process Variables
Data Summary	Missing Values	Summarize Data
Output Results	Output Type	Output Results



In= SAS Data Set Option in R Merges

SAS

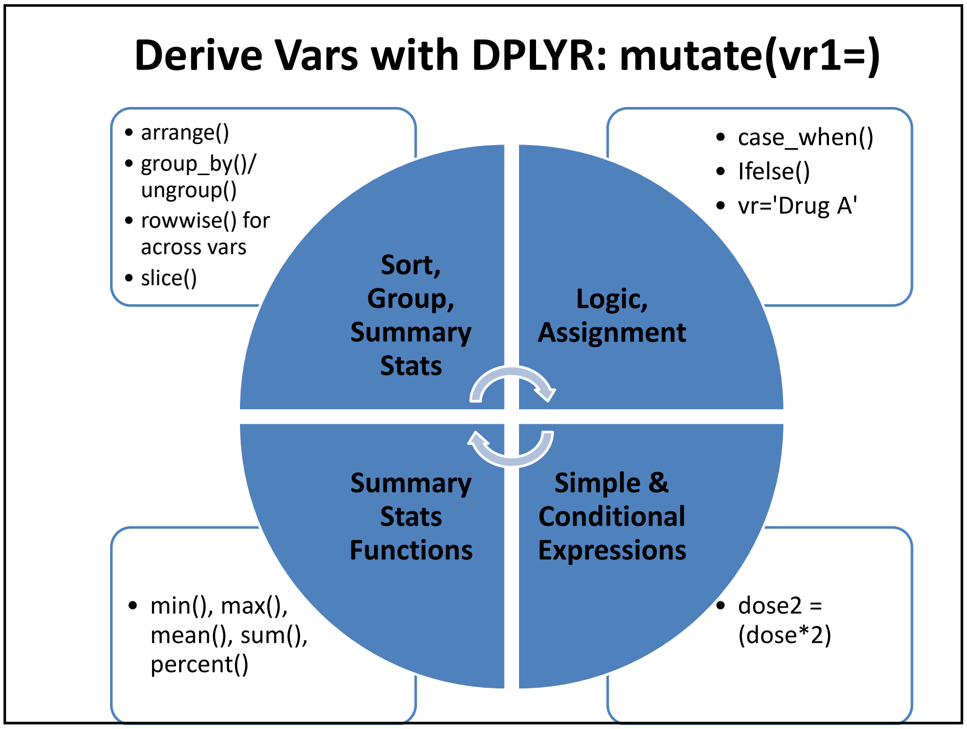
```
data df3;
merge
  df1 (in=a)
  df2 (in=b);
  by team;
 run;
```

R

```
df1 < data.frame()
df2 < data.frame()
df3 < merge(df1, df2)
```

R Functions

```
df1 <- data.frame(df1, dsn='df1')
df2 <- data.frame(df2, dsn='df2')
df3 <- merge(df1, df2, by=c('team'), all.x=TRUE, all.y=TRUE)
```



Derive Vars with DPLYR: mutate(vr1=)

Sort, Group, Summary Stats

- arrange()
- group_by()/ ungroup()
- rowwise() for across vars
- slice()

Logic, Assignment

- case_when()
- Ifelse()
- vr='Drug A'

Summary Stats Functions

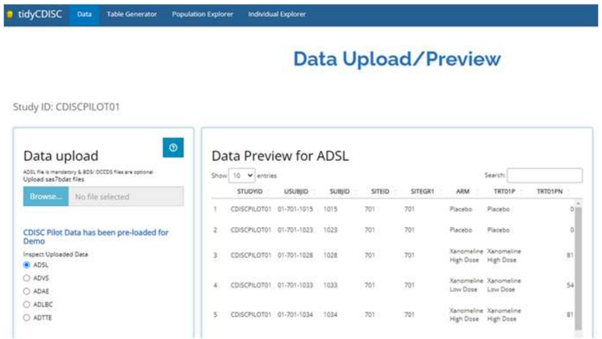
- min(), max(), mean(), sum(), percent()

Simple & Conditional Expressions

- dose2 = (dose*2)

What about pharmaverse Shiny Apps? tidyCDISC demo

Metadata > OAK > Admiral > Define.xml > TLGs (rtf/pdf) > Submissions > **Shiny**



Data Upload/Preview

Study ID: CDISCPILOT01

Data upload

ADSL file is mandatory & BDS/ OCCDS files are optional
Upload sas7bdat files

Browse... No file selected

CDISC Pilot Data has been pre-loaded for Demo

Inspect Uploaded Data
● ADSL
○ ADVS
○ ADAE
○ ADLBC
○ ADTTE

Data Preview for ADSL

Show 10 entries Search:

	STUDYID	USUBJID	SUBJID	SITEID	SITEGR1	ARM	TRT01P	TRT01PN
1	CDISCPILOT01	01-701-1015	1015	701	701	Placebo	Placebo	0
2	CDISCPILOT01	01-701-1023	1023	701	701	Placebo	Placebo	0
3	CDISCPILOT01	01-701-1028	1028	701	701	Xanomeline High Dose	Xanomeline High Dose	81
4	CDISCPILOT01	01-701-1033	1033	701	701	Xanomeline Low Dose	Xanomeline Low Dose	54
5	CDISCPILOT01	01-701-1034	1034	701	701	Xanomeline High Dose	Xanomeline High Dose	81

Experience tidyCDISC App Now:

https://rinpharma.shinyapps.io/tidyCDISC

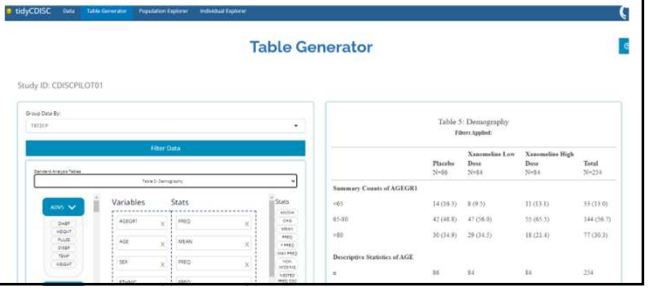


Table Generator

Study ID: CDISCPILOT01

Group Data By: TRT01P

Filter Data

Standard Analysis Tables: Table 5: Demography

Variables	Stats
AGEGR1	FREQ
AGE	MEAN
SEX	FREQ
ETHNIC	FREQ

Table 5: Demography
Filters Applied:

	Placebo N=86	Xanomeline Low Dose N=84	Xanomeline High Dose N=84	Total N=254
Summary Counts of AGEGR1				
<65	14 (16.3)	8 (9.5)	11 (13.1)	33 (13.0)
65-80	42 (48.8)	47 (56.0)	55 (65.5)	144 (56.7)
>80	30 (34.9)	29 (34.5)	18 (21.4)	77 (30.3)
Descriptive Statistics of AGE				
n	86	84	84	254

Empower your Team
- Data Review
- Table Generator
- Population Explorer
- Individual Explorer

SAS and R

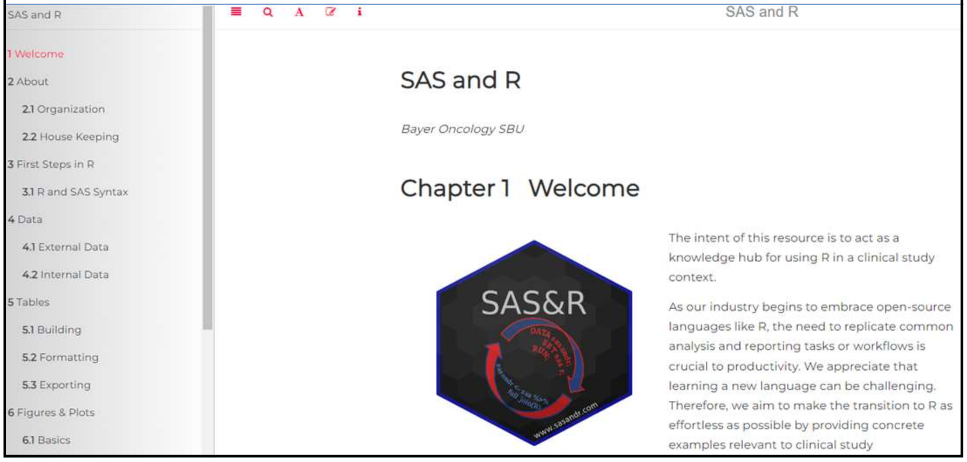
https://r4csr.org, https://bayer-group.github.io/sas2r

R for Clinical Study Reports and Submission

Delivering TLFs in CSR > 1 Overview

1 Overview

1.1 Background



SAS and R

1 Welcome
2 About
2.1 Organization
2.2 House Keeping
3 First Steps in R
3.1 R and SAS Syntax
4 Data
4.1 External Data
4.2 Internal Data
5 Tables
5.1 Building
5.2 Formatting
5.3 Exporting
6 Figures & Plots
6.1 Basics

SAS and R

SAS and R

Bayer Oncology SBU

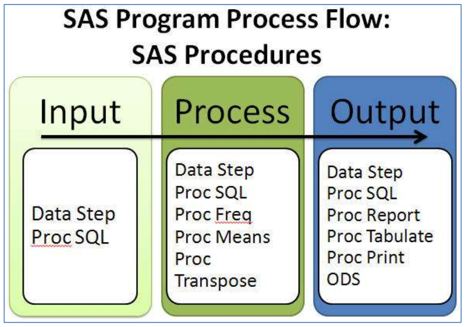
Chapter 1 Welcome

The intent of this resource is to act as a knowledge hub for using R in a clinical study context.

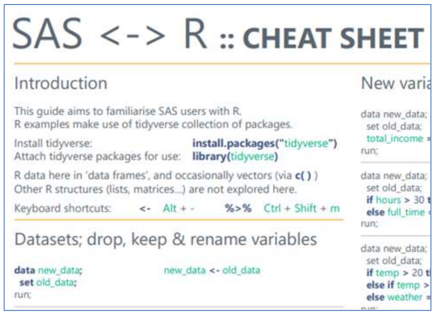
As our industry begins to embrace open-source languages like R, the need to replicate common analysis and reporting tasks or workflows is crucial to productivity. We appreciate that learning a new language can be challenging. Therefore, we aim to make the transition to R as effortless as possible by providing concrete examples relevant to clinical study

R-Guru: Compare with SAS

https://r-guru.com/Compare-with-SAS



SAS Program Process Flow:
SAS Procedures



SAS <-> R :: CHEAT SHEET

Introduction

This guide aims to familiarise SAS users with R.
R examples make use of tidyverse collection of packages.

Install tidyverse: install.packages("tidyverse")
Attach tidyverse packages for use: library(tidyverse)

R data here in 'data frames', and occasionally vectors (via c( ))
Other R structures (lists, matrices...) are not explored here.

Keyboard shortcuts: <- Alt + - %>% Ctrl + Shift + m

Datasets; drop, keep & rename variables

```
data new_data;
 set old_data;
run;
```

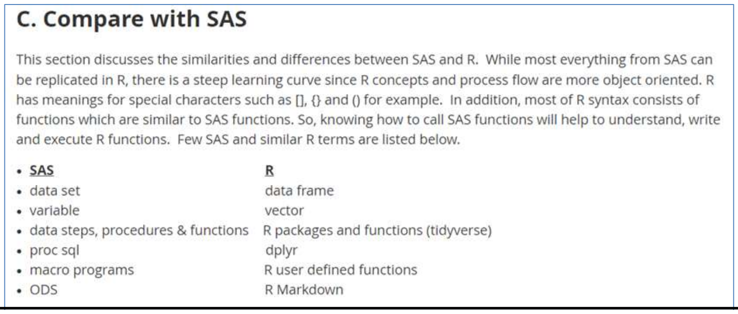
```
new_data <- old_data
```

New varia

```
data new_data;
 set old_data;
 total_income =
run;

data new_data;
 set old_data;
 if hours > 30 t
 else full_time =
run;

data new_data;
 set old_data;
 if temp > 20 t
 else if temp >
 else weather =
```



C. Compare with SAS

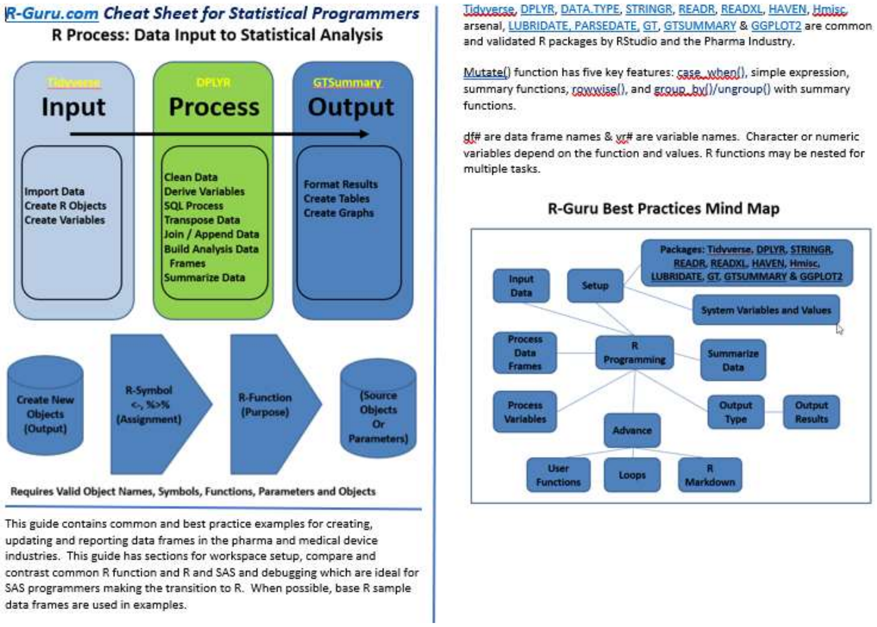
This section discusses the similarities and differences between SAS and R. While most everything from SAS can be replicated in R, there is a steep learning curve since R concepts and process flow are more object oriented. R has meanings for special characters such as [], {} and () for example. In addition, most of R syntax consists of functions which are similar to SAS functions. So, knowing how to call SAS functions will help to understand, write and execute R functions. Few SAS and similar R terms are listed below.

SAS	R
data set	data frame
variable	vector
data steps, procedures & functions	R packages and functions (tidyverse)
proc sql	dplyr
macro programs	R user defined functions
ODS	R Markdown

27

R-Guru Cheat Sheet is for SAS Programmers

Download at R-Guru.com



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.

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