

Paper TT09

R Evolution: A Guide for Training SAS Programmers

Introduction

Introduction

Katalyze Data (formerly Amadeus Software). A boutique data and analytics consultancy focused on expert advice and bespoke implementations to unlock the power of your data.

We have delivered SAS training for over 30 years, and over the last few years have developed and delivered R training, often to SAS users who tend to have similar issues and concerns.

Today – we will distill those lessons to help you on your own training journeys!



Contents

01

Methodologies
Of Training R

02

Common Issues
and Solutions

03

Ways to
Integrate R and
SAS

04

Conclusion and
Recommendations

Methodologies of Training R

Preparation and Considerations

1. Know Your Audience

- Find engaged and ready SAS users
- Survey to discover use-cases and skills

2. Incentivise Learning

- Address pain points
- Encourage competition

3. Support and Community

- Develop mentorship programs
- Ongoing support and community of practice



How To Train SAS Programmers in R

1. Workshop Based Learning

- Instructor-led sessions
- Self-paced additional learning

2. Project Based Learning

- Real-life projects
- Comparative coding

3. Course Based Learning

- Online Courses

4. Gamification, Continuous Learning, and Collaboration



What To Teach SAS Programmers



- **Fundamental concepts vs task-based learning**
- **Choosing (or not) what to teach – base R and the tidyverse**
- **Specialisations and advanced techniques**

An Example Content List

- The main data types and structures in base R, how to manipulate them using a set of basic functions and operators
- Simple conditional statements and loops
- Reading & writing standard data formats
- Using the core tidyverse packages for data manipulation and visualisation
- The 6 main {dplyr} verbs and how to use them
- Simple string manipulations, with {stringr}
- Datetime processing with {lubridate}
- Plotting with {ggplot2}
- User-defined functions
- Basic statistical analysis with base R
- More specialised statistical analysis using various packages (dependent on the user's role)

Common Issues and Solutions

When Teaching R To SAS Programmers

Thinking About Data

- Datasets, compared to vectors and data.frames
- Data Steps and Procedures
- Objects, functions, and more objects



Thinking About Data

```
data have;  
  do i=1 to 25;  
    x = rand("uniform");  
    output;  
  end;  
  drop i;  
run;  
  
data want;  
  set have;  
  y + x;  
run;  
  
proc print data=want(obs=3); run;
```

Obs	x	y
1	0.32510	0.32510
2	0.63831	0.96341
3	0.79534	1.75875



```
have <- runif(25)  
  
want <- double(length(have))  
tot <- 0  
  
for (i in seq_along(want)) {  
  want[i] <- tot <- tot + have[i]  
}  
  
head(want)  
[1] 0.6675946 1.2336240 2.1378750 3.0989581  
3.6510822 4.3223310
```

Iterative vs Vectorised

```
have <- runif(25)

want <- double(length(have))
tot <- 0

for (i in seq_along(want)) {
  want[i] <- tot <- tot + have[i]
}

head(want)

[1] 0.6675946 1.2336240 2.1378750 3.0989581
3.6510822 4.3223310
```



```
want <- cumsum(have)
head(want)

[1] 0.6675946 1.2336240 2.1378750 3.0989581
3.6510822 4.3223310
```

Missing Values - SAS

- Missing in SAS, character and numeric
- Operations and comparisons on missing values in SAS

```
data _null_;  
  x = "";  
  y = "value";  
  z = x || y;  
  put z=;  
run;  
z=value
```

```
data _null_;  
  x = .;  
  y = 42;  
  z1 = x + y;  
  z2 = sum(x, y);  
  put z1= z2=;  
run;  
z1= . z2=42
```

```
data _null_;  
  x = .;  
  y = -4.2e100;  
  if x < y then put "True";  
  else put "False";  
run;  
True
```

Missing Values - R

```
x <- c(1, 5, NA, 2)
y <- c(0, NA, 4.5, 2)

x + y
[1] 1 NA NA 4
```

```
x <= y
[1] FALSE    NA    NA    TRUE
```

```
rowSums(cbind(x, y), na.rm = TRUE)
[1] 1.0 5.0 4.5 4.0
```

Big Data

- Big data in SAS – how is it handled
 - On-disk
 - Occasionally in-memory
 - The same techniques any size
- Big data in R, a trickier story
 - In-memory
 - Different techniques for big data



Big Data in R – 3 Approaches

```
read_csv("data/big_data.csv", lazy = TRUE) |>  
  filter(x == 1) |>  
  summarise(mean = mean(y))  
  
# A tibble: 1 × 1  
  mean  
  <dbl>  
1 0.500
```

Big Data in R – 3 Approaches

```
library(duckplyr)

df_from_csv("data/big_data.csv") |>
  filter(x == 1) |>
  summarise(mean = mean(y))

# A tibble: 1 × 1
  mean
  <dbl>
1 0.500
```

Big Data in R – 3 Approaches

```
library(bigmemory)

bm <- read_csv("data/big_data.csv") |>
  as.matrix() |>
  as.big.matrix(backingfile = "big_data.bin", backingpath = "data/bigmemory")

mean(bm[bm[, "x"] == 1, "y"])

[1] 0.4998729
```

Different Numerical Results

```
head(x)
[1] 91.5 93.7 28.6 83.0 64.2 51.9
cat("Total is:", sum(round(x)))
Total is: 50013540
```

VS

```
data tot;
  infile "x.csv" end=end;
  input x;
  y + round(x);
  if end then
    put "Total is: " y;
run;
Total is: 50063493
```

<https://psiaims.github.io/CAMIS/>

Ways to Integrate R and SAS

Using R in SAS & SAS in R

Using the command line from SAS and R

```
x """"%sysget(R_HOME)/bin/Rscript"" test.R";
```

```
rc = system2("sas", args = c("-sysin", "myprogram.sas"))
```

This does require boilerplate and careful consideration of inputs/outputs

Using R in SAS & SAS in R

Using PROC IML in SAS, to directly execute R code

```
proc iml;  
  call ExportDataSetToR("sashelp.cars", "df");  
  
  submit / R;  
    source("myscript.R")  
    df2 <- myfunction(df)  
  endsubmit;  
  
  call ImportDataSetFromR("df2", "cars2");  
quit;
```

Using R in SAS & SAS in R

Using {swat} in R to communicate with CAS in SAS Viya

```
library(swat)
conn <- CAS('hostname')
cars_cas <- as.casTable(conn, mtcars)

mean(cars_cas$mpg[cars_cas$mpg > 20])

cas.dataStep.run(conn, code = "data cars; set cars; mpg = mpg + 1; run;")
```

Operating on a SAS
dataset using R
syntax!

Using R in SAS & SAS in R

Using SASPy via Python with {reticulate} in R

```
library(reticulate)
saspy = import("saspy")
sas = saspy$SASsession()
cars = sas$sasdata("cars", "sashelp")

cars_filt = cars$where("mpg > 20")$sort("mpg", out = "cars_filt")

sas$submit("proc means data=cars_filt; run;")
```

Conclusion and Recommendations

Conclusion and Recommendations

A checklist for designing one kind of training plan:

- ❑ Start small, with enthusiastic SAS programmers who will use R immediately
- ❑ Design a training plan that starts with primarily instructor led workshops.
- ❑ Include exercises replicating day-to-day tasks
- ❑ The content should cover both base R, and some key packages (such as the tidyverse).
- ❑ Complement the workshops with self-paced learning materials
- ❑ Build a central community of practice
- ❑ Design a mentorship program that pairs experienced R users with SAS programmers
- ❑ Ensure continued support post-training through this support network

References & More

We are exhibiting in the main conference center – come and say hello!



github.com/michaelwalshe/phuse-2024-r-evolution

Cetinkaya-Rundel, Mine, Johanna Hardin, Ben Baumer, Amelia McNamara, Nicholas Horton, and Colin Rundel. 2022. “An Educator’s Perspective of the Tidyverse.” *Technology Innovations in Statistics Education* 14 (1). <https://doi.org/10.5070/t514154352>.

Matloff, Norman. 2023. “TidyverseSkeptic.” <https://github.com/matloff/TidyverseSkeptic>.

Min-Hua Jen, Kyle Lee, Brian Varney. 2023. “Key Considerations When Understanding Differences in Statistical Methodology Implementations Across Programming Languages – an Introduction to the CAMIS Project.” <https://phuse.s3.eu-central-1.amazonaws.com/Deliverables/Data+Visualisation+%26+Open+Source+Technology/WP077.pdf>.

Pharmaverse. 2024. “Pharmaverse.” <https://pharmaverse.org/>.

R Consortium. 2024. “Using r to Submit Research to the FDA: Pilot 4 Successfully Submitted to FDA Center for Drug Evaluation and Research.” <https://r-consortium.org/posts/using-r-to-submit-research-to-the-fda-pilot-4-successfully-submitted/>.

