**Paper TT09**

# R Evolution: A Guide for Training SAS® Programmers

Michael Walshe, Katalyze Data, Oxford, United Kingdom

## Abstract

In the changing landscape of pharmaceutical programming, many companies are exploring R alongside SAS for its open-source flexibility and broad community. However, some SAS programmers find it difficult to transition to using R, and it often falls to the next generation of statistical programmers to foster and encourage the use of R in their company. Based on practical experience designing and delivering R training to SAS programmers, this presentation dives into how to train SAS programmers in R, highlighting effective methodologies, resources, and best practices tailored for those familiar with SAS. It outlines common pitfalls and challenges that SAS programmers may encounter when learning R, such as: different modes of thinking about data; dealing with missing values; changes in numerical methods; and dealing with larger-than-memory data. This presentation offers practical solutions and strategies to overcome these hurdles, and how to incorporate them into a training program designed for SAS programmers.

## Introduction

SAS and Pharma is a tale as old as time, but just a brief scan over the topics at this year's PHUSE EU Connect reveals that it's no longer the only story, Pharma is a multilingual industry now more than ever.

SAS has been the dominant language in the industry for decades, its robust and well-tested suite of procedures as well as the support offered have made it a trusted tool for handling the critical data that whole industries rely on *[ref here]*. Many organisations have established processes and procedures that are built around SAS, solidifying its place within industry workflows. R, however, has seen increasing adoption in recent years, particularly as the pharmaceutical industry embraces the open-source movement and starts to realise the benefits of the flexibility and broad community that R offers. This includes a vast array of packages and tools written by the users (including pharma companies themselves *[ref here]*), and recently whole submissions have been performed in R *[ref here]*.

The coexistence of SAS and R within organisations presents an opportunity for collaboration, rather than just competition. For SAS programmers, learning R offers the chance to expand their analytical tool-kit and take advantage of alternative visualisation tools such as Shiny. For R programmers, learning SAS can provide a deeper understanding of the data management and regulatory requirements that are critical in the pharmaceutical industry, as well as the opportunity to work with and learn from the wealth of software created by SAS programmers. Today, we're going to focus on the first of these opportunities, and explore how SAS programmers can successfully start using R productively. We will first discuss how to design a training plan for SAS programmers, and then cover common issues and solutions that SAS programmers may encounter when learning R, before finally outlining a check-list of recommendations for a successful training program.

## Considerations and Design Of A Training Plan

Transitioning SAS programmers to R in a structured and effective manner requires careful planning. Designing a training program involves addressing not only the technical differences between SAS and R but also how to manage organisational and user buy-in, ensuring the transition is smooth and successful. This should not be a training plan that is a one-off, but rather a continuous approach to learning and development that starts by training the SAS developers most eager to learn R, and then makes these developers the champions of R within the organisation. In this section, we'll discuss the various considerations for designing a training plan, how to get to a point in your organisation where you can begin to roll-out a training plan, and the different types of training plans and teaching methods.

#### Getting Organisation and User Buy-In

The first step for any successful transformation program at an organisation is to make sure that the goals and process is fully understood by both the organisation at large, and the users who will have to enact the change.

**Securing Organisational Buy-In**

Organisational buy-in is crucial, and in most Pharmaceutical companies the benefits of R are already well understood. However, it's important to make sure that all stakeholders are on board, and that the training program you have designed is aligned with the company's strategic goals. To secure organisational buy-in:

*1. Demonstrate Value*

- Cost benefits – R is not necessarily cheaper than SAS when the total cost of platforms, migration, hiring, and more is considered [ref. paper on cost and time to value]. However, there can be economies and cost efficiencies. Outline these if possible.
- Alignment with innovation – Explain how R's use in academia and research, and its rapid innovation in specialized fields such as genomics and machine learning, can drive new opportunities for the organisation. New graduates are more likely to be proficient in R, and the current explosion in Pharma specific R packages means that to take advantage of new techniques and tools, R is a must.

*2. Collaborate with Leadership*

- Have a champion in the senior leadership team – With any transformation, there must be a champion at the top. Explain the value of the training plan you've developed to them, and use their support to drive the change.
- Feedback on success – Measure success by tracking key performance indicators such as reduced analysis time, improved accuracy, or feedback from participants. Then circulate this with the leadership and beyond to show the value of the training program.

**Securing User Buy-In**

Perhaps more important than organisational buy-in is user buy-in. If the users are not engaged with the training program, it is unlikely to be successful. Moreover, some users can be actively hostile to change if they feel it is not beneficial or threatening to them. To avoid this, instead first target the early adopters and help them become proficient in R. These users should have an immediate use-case that is high-impact or a pain-point in the SAS process. Once some high-impact use-cases have been demonstrated, the rest of the organisation will be more likely to follow. [ref - early/most,late graph]. You may already know who these early adopters are - if not, a survey or a simple question to team leaders can help you identify them. Other ways of encouraging users are:

*1. Incentivise Learning*

- Encourage competition – Create a friendly competition among users to motivate them to learn R. Offer rewards or recognition for completing training modules or achieving proficiency in R, and host hackathon-style events to showcase their skills.
- Highlight career benefits – Show how learning R can enhance career prospects, with dual proficiency making programmers more versatile and competitive in the market. Certifications are less used in R compared to SAS, but can be a motivator for some users and demonstrate proficiency to others.
- Address pain points – R can be intimidating for SAS users due to differences in syntax and structure. Be upfront about the challenges while emphasising how it will make their daily work more productive and rewarding. Explain how learning R will enhance certain workflows, from visualisation, to data wrangling, to machine learning and modelling.

*2. Ensure Support and Community*

- Create mentorship programs: Pair experienced R users with SAS programmers to create a collaborative learning environment, encouraging peer learning and discussion. These mentors & mentees can be at any stages in their careers, and often it is more beneficial to have a more senior SAS user alongside a junior R user. This can help to break down barriers and ensure that the learning is two-way, with the SAS user learning R, and the R user learning domain knowledge. Ideally, these pairs are not only learning together but are working on active projects together as well.
- Ongoing support – Ensure continued support post-training through forums, Q&A sessions, and reference guides. Providing a help desk or dedicated team to assist during the transition can ease the learning curve as well.

**Considerations for Designing A Training Plan**

Once you have the go-ahead to begin the training and transformation, and are starting to design a training plan for the SAS programmers in your organisation, you first need to consider the following:

*1. Assess Skill Levels and Engagement*

SAS programmers come with strong statistical and data manipulation skills, but they may lack familiarity with the programming paradigms in R. Assess their current skill level through surveys, interviews, or small tests to understand

what gaps need to be filled. This can be as simple as a survey sent to all SAS users, and you should take the opportunity to also assess which users are most eager to learn R and identify your early adopters.

*2. Align Training with User Use-Cases*

The training should focus on replicating common tasks that SAS programmers face in their day-to-day roles, and so you need to understand what tasks users tackle day to day (and which are particular pain points). For example, recreating statistical procedures like `PROC MEANS` or `PROC GLM`. These should be hands on exercises as part of any training, and it is important that they aren't learning new techniques alongside new tools - so use user interviews to decide how to best target the training and exercises.

*3. Determine Frequency and Structure*

Perhaps the most important decision is how to structure the training. This can be in-person, virtual, self-paced, or a combination of these. The training could also be project-based, all in a few sessions, or spread out over weeks or months. There are pros and cons to each that we will discuss in the next section, however the choice will partly depend on the resources available to you, and the urgency of the need to train users. If you have a small group of users, eager to learn, who have an immediate use-case, then a few in-person sessions may be the best way to get them up to speed quickly while answering any questions they have. However, you may need to consider a more self-paced approach for a larger group of users, or if the users are spread out across different locations and time-zones.

**Different Methodologies for Training SAS Programmers in R**

**How to Train SAS Programmers in R**

There are several methodologies for training SAS programmers in R, each with its own strengths and weaknesses. The choice of methodology will depend on factors such as the size of the group, the urgency of the need to train users, and the resources available. As such, there is no single method that is best for all situations, and a combination of methods may be most effective.

*1. Workshop Based Learning*

- Instructor-led sessions – Start with interactive instructor-led workshops that introduce the basics of R, its syntax, and some essential libraries. These can be in-person or virtual, but should be highly interactive, encouraging participants to ask questions and with activities throughout. This is one of the best ways to encourage engagement, however requires a time investment.
- Self-paced e-learning – Workshops should be complemented with self-paced learning materials, such as internal documentation, cheat-sheets, and user-guides. These resources can help reinforce learning and provide a reference for users as they start using R in their daily work. Some users can learn only using self-paced resources, however these are in the minority.

*2. Project Based Learning*

- Real-life projects – Use pharmaceutical datasets for project-based learning to simulate real-world challenges. This method helps participants immediately apply their learning, solidifying their understanding of R. Projects can be tailored to the needs of the organisation, and workshops can be delivered to teach users just enough to complete the next set of tasks before re-convening to share and discuss before the next step. These will require more resources and time to set up, but can be more effective than traditional sessions.
- Comparative coding – Include tasks where learners convert existing SAS code to R, potentially from simplified production code. This can build confidence and reinforce how R handles common pharmaceutical industry tasks, such as data cleaning, statistical analysis, and reporting.

*3. Course Based Learning*

- Online courses – Use online platforms like Coursera, DataCamp, or Udemy to provide structured courses that cover R basics and advanced topics. These platforms offer flexibility and self-paced learning, allowing users to learn at their own pace. However, they may lack the interactivity and engagement of instructor-led workshops. Additionally, they may not be tailored to the specific needs of the organisation.

*4. Gamification, Continuous Learning, and Collaboration*

In addition to the 3 main approaches outlined above, these can all be enhanced using the following techniques:

- Gamify the learning experience – Introduce quizzes, challenges, or coding competitions that encourage continuous learning and reinforce the material. Rewards or recognition can be incorporated to boost motivation.

- Microlearning – Use short modules (10-15 minutes) that cover specific topics like R's ggplot2 or handling clinical trial data in R. This can help maintain momentum and prevent overwhelming the users, and can provide great reference or refresher material.
- Pair programming and group exercises – Encourage team-based learning activities where programmers work together to solve problems. This promotes collaboration and accelerates learning by allowing users to share their expertise.

#### What to Train SAS Programmers in R

There are two main schools of thought for teaching a new programming language to those who are already proficient in another. The first is to start with fundamental concepts in the language, ensure a solid understanding, and gradually introduce more complex topics or techniques. The second is to teach just enough to get started with a task, and then introduce more topics as they become necessary. Both the task-based and concept-based approaches have their merits, and most appropriate will depend on the audience, goals, and delivery method of the training. For example, if the training is self-paced, a task-based approach may be more effective, as it allows learners to immediately apply their knowledge and ensures that after each block of training a learner will have gained some actionable piece of knowledge. On the other hand, an instructor-led workshop may benefit from a concept-based approach, as you can ensure that all learners have a solid foundation while still providing set opportunities for hands-on practice and ensuring that at the end of the workshop all learners know enough concepts to be productive day-to-day. Whichever method you use, it's important when teaching SAS programmers R that you include resources to help them translate concepts they are familiar with, and understand where these analogies break down.

Specific to R, when teaching you may think you need to decide which "style" of R to teach. Although it isn't as pronounced a difference as some like to make it seem, there is a split in R programming styles. The first are those who are primarily "tidy" coders, using packages from the tidyverse or that are similar in style. The alternative is sometimes just defined in opposition (i.e. "non-tidy") but can be seen as more consistent with base R, making more use of loops and operators such as `[` and `$`.

The fallacy here is that you have to choose. Each approach has its benefits and drawbacks, but we believe that a comprehensive R education must cover both. The tidyverse is very consistent, generally user-friendly, and has a large number of packages that are designed to work together. More-over, users are often more productive faster with the tidyverse, though claim has been challenged. Base R, by contrast, is the language at is purest form, and understanding it is essential to understanding how R works. It is also very flexible, and can be more performant in some cases *[ref here]*.

To be optimally productive in R, we have found that a SAS user should, by the end of a full training program, be familiar with the following concepts:

- The core data types and structures in base R (i.e. vectors, lists, data.frames)
- How to do standard manipulations on these (subsetting, filtering, and aggregating).
- A selection of core functions in base R
- Simple conditional statements and loops
- Reading & writing standard data formats
- Using the core tidyverse packages for data manipulation and visualisation
- The main dplyr verbs and how to use them
- Simple string manipulations
- Date/time processing with `{lubridate}`
- Plotting with `{ggplot2}`
- User-defined functions
- Basic statistical analysis with base R
- More advanced statistical analysis using various packages (dependent on the user's role)

This is a lot to cover, and so it is important to be selective in what you cover in each session, and to ensure that you are providing resources for users to continue learning and refer back to after sessions are complete. These will then have covered most use-cases in SAS, including data steps, common statistical procedures, PROC SQL, ODS graphics, macros, and more.

## Teaching R to SAS Programmers – Common Issues and Solutions

Whichever method is ultimately chosen to train SAS programmers in R, there will be some common issues that arise. SAS is a powerful language, but is also very different to many more modern languages, and so there are some key differences that can trip up users. In this section, we will discuss some of the common issues that SAS programmers may encounter when learning R.

**Thinking About Data**

In SAS the essential unit of data is the *dataset*, comprised of *observations* and *variables*. In R, the essential atomic structure is the `vector`, a homogeneous sequence of elements. This is the first difference that SAS users must become accustomed to, in SAS although you will frequently operate on a single variable, that will always be within the context of a dataset. In R, you will frequently operate on a single vector, although more often you will be working with a `data.frame`, a tabular data structure with *rows* and *columns* (occasionally observations and variables, see `str(df)`) this is analogous to our *dataset*.

A key difference in R is that you aren't always operating in the context of a data step or procedure. In SAS, every operation is preceded by a statement that will declare which step you are using – either a procedure, that can be statistical or more general such as `PROC SQL`, or a data step that will directly operate on and manipulate a dataset. In R there are only variables (pointing at objects/data) and functions that operate on those variables, and these are combined to form a script that will execute in order. The way of working in R is more typical of most programming languages, so SAS users who have experience in other languages (or even just with `PROC IML`) will be familiar, but for those who aren't spending some time on understanding what are variables, objects, and the context of running a script is essential. There should be time set aside in any course to explain and familiarise users with these concepts - in particular the idea that all work in R will be done in scripts with functions operating on objects.

Even basic operators in R are really functions, for example:

```
`+`(1, 2)

[1] 3
```

However this is a topic to be introduced only when covering advanced topics in R.

Once a SAS programmer is used to these first differences with R, the next key hurdle to overcome is the different way of natively expressing operations in R. Whereas in SAS you will typically use a data step, and so think of the operation in terms of the data step's implicit loop over a dataset (via our `LPDV`), in R you will typically use a function that will operate on a vector or data frame in its entirety. Although in the implementation of these functions there will be a loop somewhere, this is normally in the implementation in C or another low-level language, and is hidden from the user in the name of efficiency. This efficiency means that to write an operation on data in R, you should express it in terms of R's built-in functions and operations rather than writing a loop yourself.

For example, in SAS to calculate a cumulative sum you can do:

```
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=5); run;

 Obs       x          y

   1    0.32510    0.32510
   2    0.63831    0.96341
   3    0.79534    1.75875
   4    0.81953    2.57828
   5    0.99284    3.57112
```

In R, if you tried to replicate this naively you could write:

```
have <- runif(25)
want <- double(length(have))
tot <- 0
for (i in seq_along(want)) {
```

```
  want[i] <- tot <- tot + have[i]
}
head(want)

[1] 0.239036 1.050265 1.500162 2.115167 3.097275 3.339853
```

But due to the loop implemented in R this wouldn't be efficient, instead you should try to either find a function that performs this operation or express your calculation using R's *vectorised* operators and functions:

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

[1] 0.239036 1.050265 1.500162 2.115167 3.097275 3.339853
```

Understanding vectorisation in R, and taking advantage of it, is a key learning point that should be emphasised and encouraged in any training program. Users will often be encouraged to take advantage of it simply due to the fact that calculations expressed this way are often more elegant – see above – but any training should emphasise and encourage these techniques wherever possible.

#### Missing Values

Missing values are a common factor in any data analysis. They can be cause problems, but are a key technique that allows a user to meaningfully represent data that truly is not there (because of any number of issues in surveying, data collection, or elsewhere). In R, missing values are represented by an `NA`, whereas in SAS they are either a `.`, an empty string `""`, or a special missing value (e.g. `.A`). However the key difference is how they are treated in calculations and comparisons.

In SAS, missing character values are simply treated as an empty string, and not propagated when used in operations.

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

z=value
```

Meanwhile missing numbers will be propagated in some operations, and not in others (e.g. `sum(...)`):

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

z1=. z2=42
```

And in comparisons, missing numbers are treated as negative infinity (i.e. less than any other number):

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

True
```

This is simpler in R, but can be unexpected. In R, unless explicitly stated, missing values are *always* propagated, of any type. This is most often a source of confusion in comparisons – but just adding an explicit check for missing or using the `na.rm` argument in functions can solve this:

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

x + y
```

```
[1]  1 NA NA  4
```

```r
rowSums(cbind(x, y), na.rm = TRUE)
```

```
[1] 1.0 5.0 4.5 4.0
```

```r
x <= y
```

```
[1] FALSE    NA    NA  TRUE
```

When teaching missing values in R, take care to emphasise the differences in how they operate, as well as the similarities. Techniques for dealing with them should be covered as new tools are introduced.

#### Big Data

One issue that can trip up SAS programmers as they transition to R is dataset size limitations. In SAS, for most operations a dataset of nearly any size can be manipulated, so long as you have enough disk space. This is because only a portion of the data is in-memory at any one time, and the data is shuttled from disk into memory in buffered chunks. So even terabytes of data can be handled on smaller machines, it just takes more time. In R data is all stored in memory – and you may have experienced the dreaded out of memory errors, which can crash an R session and lead even the most experienced R user to despair. There is no single solution to this, however third-party packages can help alleviate the problem.

- If the size is close to the memory limit, and you will be filtering or aggregating the data, then try setting `lazy = TRUE` in `readr::read_csv` to not read in all the data at once. Alternatively, use `{data.table}` and its alternative reference semantics (e.g. in `DT[, col := value]`) to use less memory when working with your data.
- However if the data is much larger than memory, then try using the `{duckplyr}` and `{arrow}` packages, or `{dbplyr}` with another database back-end. These all let you use `{dplyr}` syntax but execute the operations in the database, only pulling the results back into memory when required. This can be more performant even if you don't have a large dataset, but depending on the backend can allow you to perform operations on datasets that are many times larger than memory.
- If you want a drop-in replacement for some operations that are more base R style, try `{disk.frame}`, though it is no longer actively developed it provides an intuitive replacement for the classic R `data.frame` that stores the data on disk.
- If you're manipulating data that is all numeric, try changing it to a matrix and working with the `{bigmemory}` or `{matter}` R packages.

All of these techniques, however, are considered advanced in R, so should be carefully introduced to the new programmer after there is a need. In particular, each of these strategies requires some change to the code, it is rarely a drop-in replacement after the data sizes has increased. This can be a source of frustration for SAS programmers who are used to being able to handle large datasets without thinking about it.

#### Different Numerical Results

Occasionally when translating a process from SAS to R, a user may notice discrepencies in the numeric results produced. These can be small, but occasionally are significant and can lead to concern or confusion.

For example - here we generate a random series of numbers between 0 and 100 with 1 decimal place in R, and save them to a file. If we round these to whole numbers and sum them, we get `50013540`.

```r
set.seed(42)
x <- round(runif(1e6, 0, 100), 1)
readr::write_csv(data.frame(x), "x.csv", col_names = FALSE)
head(x)
```

```
[1] 91.5 93.7 28.6 83.0 64.2 51.9
```

```r
cat("Total is:", sum(round(x)))
```

```
Total is: 50013540
```

If we then read this file into SAS and sum the rounded numbers, we get `50063493`:

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

Total is: 50063493
```

This is **49953** more than the R result! That is because SAS uses a different method of rounding than R, and so the results can be different for one particular case. In SAS, the `round` function always rounds 5 *away from zero* (i.e. up for positive, down for negative). However, in R it rounds to *whichever number is even*. These are both valid methods of rounding, however the user must be concious of them and understand why they occur. These can be highlighted and explained while training, and there is a project within PHUSE called CAMIS that is working to document and explain these differences in SAS, R, and Python at https://psiaims.github.io/CAMIS/. This is a community effort - so please contribute if you are able.

## Ways to Integrate R and SAS

When learning R, it can be important to remember that it doesn't need to be an either/or situation. It is very likely that SAS will still be in use at your organisation, and to make the most of both languages it's important to understand how they can be integrated.

#### Using R in SAS

If you wish to call existing R code from within SAS, then the simplest method would be to just execute it from the command line, for example:

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

However, this can mean that you need to have boilerplate code writing & reading the data in each language to pass it between the two. Instead, you can also call R functions from SAS, on your SAS data, using `PROC IML`:

```
proc iml;
  call ExportDataSetToR("sashelp.cars", "df");

  submit / R;
    source("myscript.R")
    df2 <- myfunction(df)
  endsubmit;

  call ImportDataSetFromR("df2", "cars2");
quit;
```

#### Using SAS from R

We can also go the other way. If you have some SAS code you want to run in R, you will first need to connect to a SAS server. The simplest method is to connect to a SAS Viya server using the `{swat}` package, and then you can directly manipulate data in SAS as if it was in R, and run arbitrary code and actions.

```
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;")
```

Or you can of course similarly use the command line, which is more convenient for SAS 9:

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

And if you're not adverse to using some Python, then you can use the `saspy` package to run SAS code from Python, and then call Python from R. A little convoluted, but it gets the job done – this can work with both SAS Viya and SAS 9.

```
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;")
```

Having demonstrations of one or more of these in your R training, likely only once users are more familiar with R, can be a good way to show the power of integrating the two languages and to help SAS users make use of their existing code and knowledge.

### Conclusion and Recommendations

Now that we have outlined the considerations, options, and methodologies for training SAS programmers in R, we can summarise them into a check-list of what a training program could look like. This isn't one-size-fits-all, but rather a starting point that can be customised based on the considerations and challenges demonstrated.

- Start with a small group of enthusiastic SAS programmers who have an immediate use-case for R. These will be your champions, and will help to train and enable the rest of the organisation.
- Design a training plan that starts with primarily instructor led workshops. There isn't a replacement for the productivity of a group of people learning together, and the ability to ask questions and get immediate feedback. It is even better if this can be in-person.
- The content of those workshops should be tailored to the day-to-day tasks that SAS programmers face, and should include hands-on exercises that replicate these tasks in R.
- The content should cover both base R, and some key packages (such as the tidyverse). If the users are already familiar with SQL, then use this as a bridge to `{dplyr}`
- Complement the workshops with self-paced learning materials, such as cheat-sheets, user-guides, and internal documentation. These should all form part of a central community of practice, where users can ask questions, share code, and learn from each other.
- Design a mentorship program that pairs experienced R users with SAS programmers. This can help to break down barriers and ensure that the learning is two-way, with the SAS user learning R, and the R user learning domain knowledge.
- Ensure continued support post-training through this support network. Use the initial groups of learners to help kick-start the transformation in the rest of the organisation, have them demonstrate how R is making them more productive through projects and hackathons.

By using this checklist or an alternative based on the considerations outlined, along with the methods and highlighting the common issues discussed, you can design a training program that effectively transitions SAS programmers to R and fosters a culture of collaboration and innovation within your organisation.