PharmaSUG 2026 - Paper TT-121

R Package Management in LSAF: Challenges and Solutions

Praneeth Adidela, ICON plc
Sagar Koona, ACL Digital

ABSTRACT

R is increasingly recognized as a powerful tool for statistical analysis and reporting in clinical research. The SAS® Life Science Analytics Framework (LSAF) provides an integrated system for managing, analyzing, reporting, and reviewing clinical research data. With the release of version 5.3, R is now integrated alongside SAS. However, in regulated environments, direct downloads of R packages from CRAN or other external sources are often restricted.

This paper presents a workflow for installing and managing R packages in LSAF using internal CRAN-like repositories and local libraries. It outlines key challenges and offers practical solutions for effective R package management within LSAF.

INTRODUCTION

In recent years, R has emerged as a widely adopted tool in clinical research, valued for its open-source nature, extensive support for statistical analyses, and powerful capabilities in data manipulation and visualization.

The SAS® Life Science Analytics Framework (LSAF) is a cloud-native statistical computing environment designed to support the complete clinical data processing workflow, including data management, analysis, reporting, review, and regulatory submission. Starting with version 5.3, LSAF introduced R integration, enabling users to develop and execute code in either SAS or R within a flexible, open platform.

Within LSAF, R programs are developed in a dedicated R session featuring an interface similar to the SAS session. R sessions are managed in the same way as SAS sessions, and the R program editor provides syntax highlighting for improved usability. R programs (*.r), data files (*.rdata, *.rds), and log files (*.rlog) are stored in the repository and workspace in the same manner as their SAS counterparts. Running, debugging, and incorporating R programs into jobs follow the same processes as SAS, with the added capability of integrating both SAS and R programs within a single job.

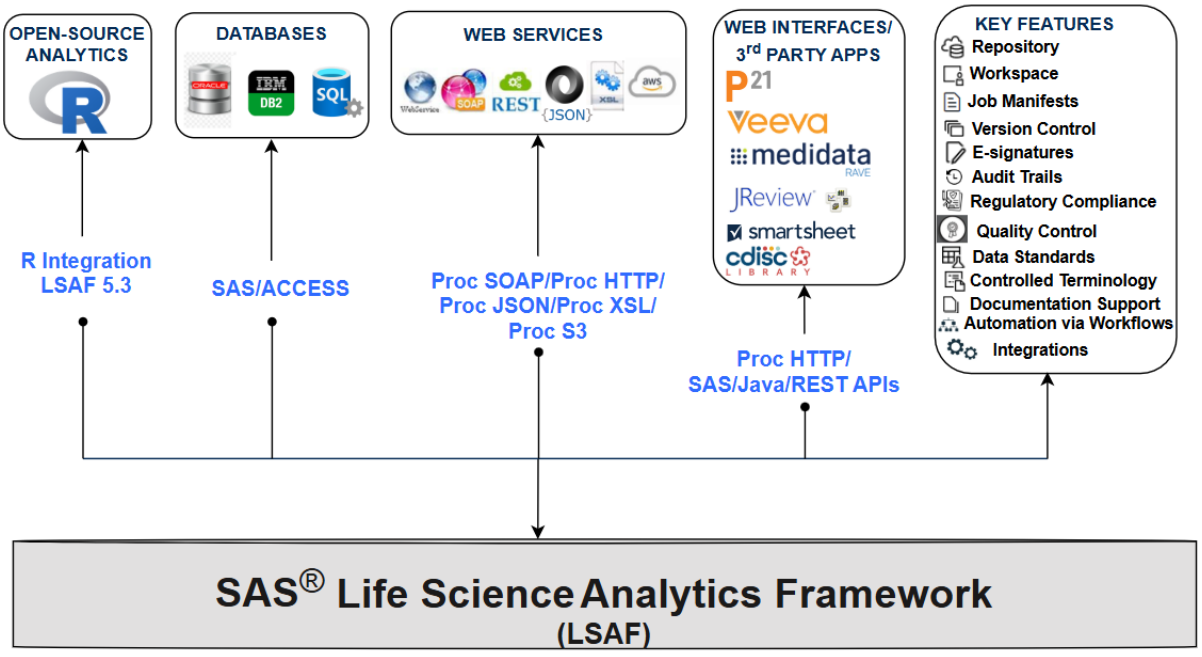


Figure 1. Overview of SAS® LSAF

R PACKAGE MANAGEMENT IN LSAF

LSAF includes a base set of pre-installed R packages and does not permit direct downloads from CRAN or other external sources. However, users can extend their analysis by setting up a local library or CRAN-like repository to install additional packages within their workspace or repository.

LSAF runs on Linux, specifically Red Hat Enterprise Linux. In this environment, the directories "/usr/lib64/R/library" and "/usr/share/R/library" serve as system-level R library directories for installing and accessing packages. The "/usr/lib64/R/library" directory contains system-provided R packages, while "/usr/share/R/library" serves as an optional shared location intended for site-wide package installations.

To install an R package, users typically use the `install.packages()` function. By default, this function attempts to install the package into the global library ("/usr/lib64/R/library"), which regular users do not have permission to write to.

The `.libPaths()` function can be used to list the directories where R searches for libraries, with the first path in the list designated as the default target for new package installations.

```
.libPaths()
```

```
> .libPaths()
[1] "/usr/lib64/R/library" "/usr/share/R/library"
```

Let's install the r2rtf package.

```
install.packages("r2rtf")
```

R attempts to install the package in the global library, but the installation fails because it lacks sufficient write access to the default system library located at (/usr/lib64/R/library), as indicated by the error message. To resolve this issue, users should create and configure a local library path that has the appropriate write permissions.

```
> install.packages("r2rtf")
Installing package into ‘/usr/lib64/R/library’
(as ‘lib’ is unspecified)
Warning in install.packages("r2rtf") :
  'lib = "/usr/lib64/R/library"' is not writable
Error in install.packages("r2rtf") : unable to install packages
```

LOCAL LIBRARY

In the LSAF system, standard users do not have permission to write to system-level libraries. Therefore, it is important to create a local library within the user's workspace or repository to install and manage custom R packages.

R_LIBS_USER is an environment variable that defines the default directory where R installs and looks for user-specific packages. The possible conversion specifiers all start with a '%' and are followed by a single letter (use '%%' to obtain '%'), with currently available conversion specifications as follows:

'%V' R version number including the patch level (e.g., '4.4.0').
'%v' R version number excluding the patch level (e.g., '4.4').
'%p' the platform for which R was built, the value of R.version$platform.
'%o' the underlying operating system, the value of R.version$os.
'%a' the architecture (CPU) R was built on/for, the value of R.version$arch.

```
R_LIBS_USER="~/R/custom-lib/%v"
Sys.getenv("R_LIBS_USER")
dir.create(Sys.getenv("R_LIBS_USER"), recursive = TRUE)
dir.exists(Sys.getenv("R_LIBS_USER"))
```

```
> R_LIBS_USER="~/Users/padidela/R/custom-lib/%v"
> Sys.getenv("R_LIBS_USER")
[1] "/lsafshared/SASWorkspaces/padidela/R/x86_64-redhat-linux-gnu-library/4.4"
> dir.create(Sys.getenv("R_LIBS_USER"), recursive = TRUE)
> dir.exists(Sys.getenv("R_LIBS_USER"))
[1] TRUE
```

The .libPaths() command now lists the local library as the first item. Therefore, subsequently installed packages will be saved in the home directory.

```
.libPaths()
```

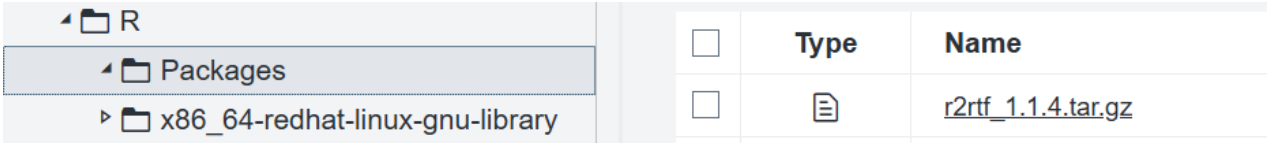
```
> .libPaths()
[1] "/lsafshared/SASWorkspaces/padidela/R/x86_64-redhat-linux-gnu-library/4.4"
[2] "/usr/lib64/R/library"
[3] "/usr/share/R/library"
```

The local library has been successfully created and is now writable, which allows for R package installation.

```
# Check write permission
#(0- Check existence, 1- Check execute permission, 2-Check write permission
and 4- Check read permission)
file.access("/lsafshared/SASWorkspaces/padidela/R/x86_64-redhat-linux-gnu-
library/4.4", mode = 2)
file.access("/usr/lib64/R/library", mode = 2)
file.access("/usr/share/R/library", mode = 2)
#Returns 0 → You have write access
#Returns -1 → You do not have write access
```

```
> # Check write permission
> #(0- Check existence, 1- Check execute permission, 2-Check write permission and 4- Check read permission)
> file.access("/lsafshared/SASWorkspaces/padidela/R/x86_64-redhat-linux-gnu-library/4.4", mode = 2)
/lsafshared/SASWorkspaces/padidela/R/x86_64-redhat-linux-gnu-library/4.4
                                                                      0
> file.access("/usr/lib64/R/library", mode = 2)
/usr/lib64/R/library
                 -1
> file.access("/usr/share/R/library", mode = 2)
/usr/share/R/library
                 -1
> #Returns 0 → You have write access
> #Returns -1 → You do not have write access
```

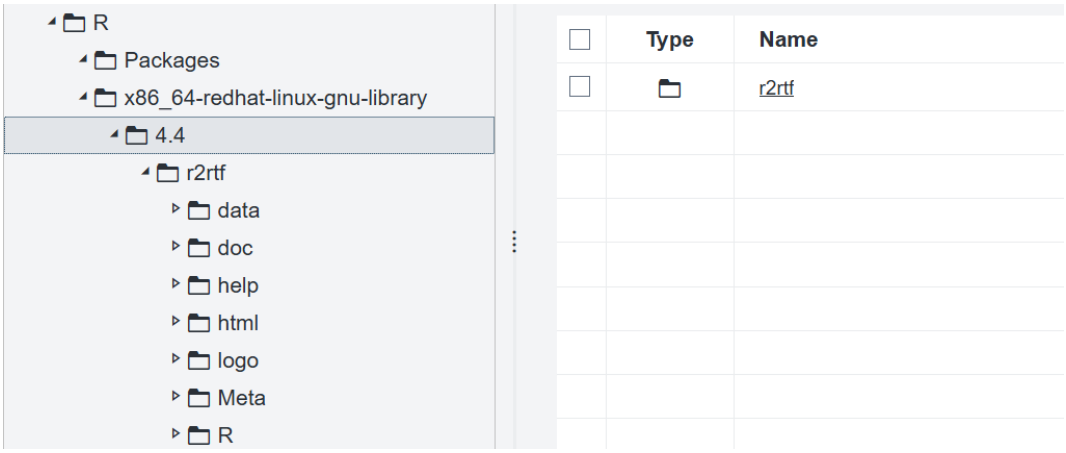
Create a folder for R packages in LSAF. Download the ‘r2rtf’ package and its dependencies to the user’s PC, then upload them to LSAF.



Install the package in the local library and call the package using the command 'library(r2rtf)'.

```
localLib <- paste0(LSAF_WS, "/R/x86_64-redhat-linux-gnu-library/4.4")
pkgPath <- paste0(LSAF_WS, "/R/Packages/r2rtf_1.1.4.tar.gz")
install.packages(pkgPath, lib=localLib, repos=NULL, type="source")
library("r2rtf")
```

After that, the new package will be installed in the local library.



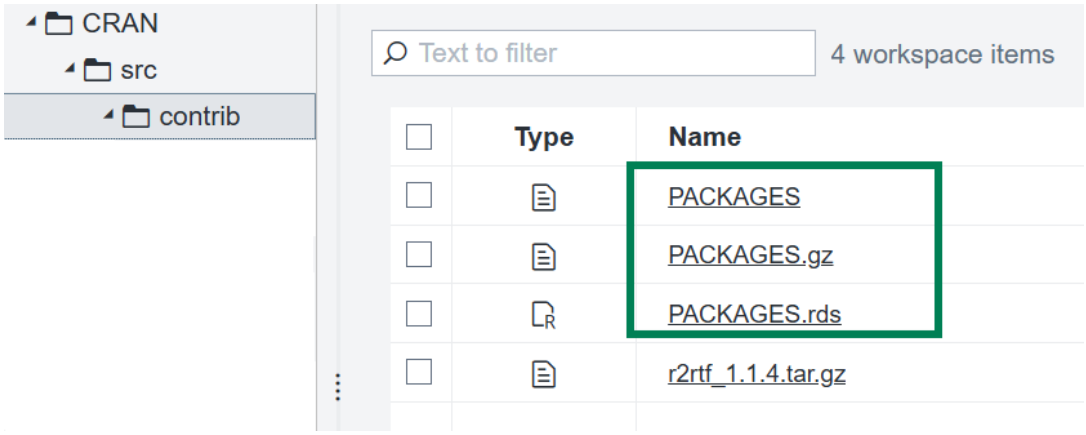
LOCAL CRAN REPOSITORY

LSAF does not permit direct downloads of packages from CRAN or other external sources. Users can set up a local CRAN-like repository within the LSAF to store selected packages for analysis.

In LSAF's Linux environment, R packages are typically installed from source (.tar.gz) files, not as binaries. The dir.create() function builds the standard src/contrib folder structure, and tools::write_PACKAGES() generates the required index files, enabling package installation in offline or restricted environments such as LSAF.

The write_PACKAGES scans the named directory for R packages, extracts information from each package's 'DESCRIPTION' file, and writes this information into the 'PACKAGES', 'PACKAGES.gz', and 'PACKAGES.rds' files. The write_PACKAGES() function only works if the directory contains R package tar.gz files.

```
repo_dir <- paste0(LSAF_WS, "/CRAN/src/contrib")
dir.create(repo_dir, recursive = TRUE, showWarnings = FALSE)
tools::write_PACKAGES(repo_dir)
```



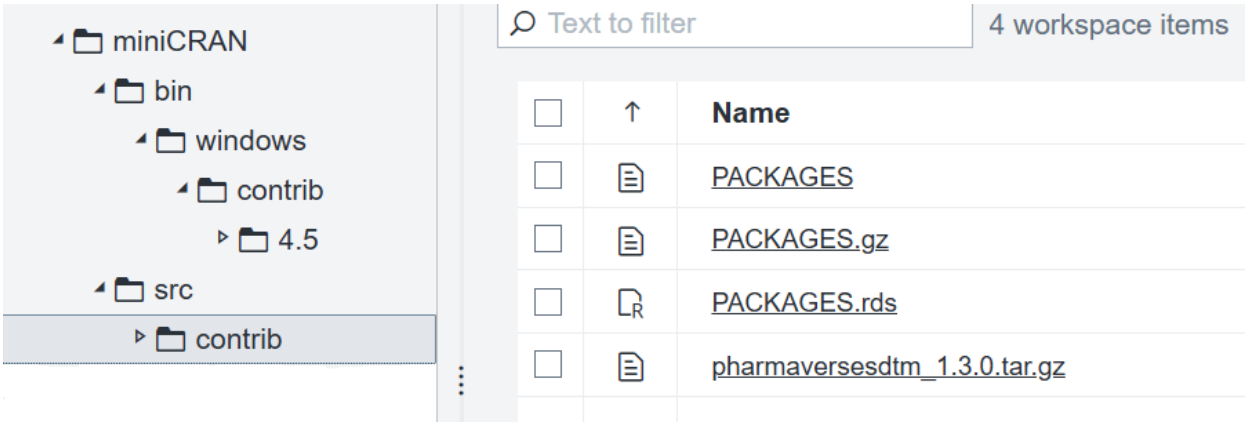
Download required packages and dependencies onto the user's PC and upload them to the LSAF.
The functions `install.packages()` and `uninstall.packages()` work effectively. Package dependencies are handled automatically.

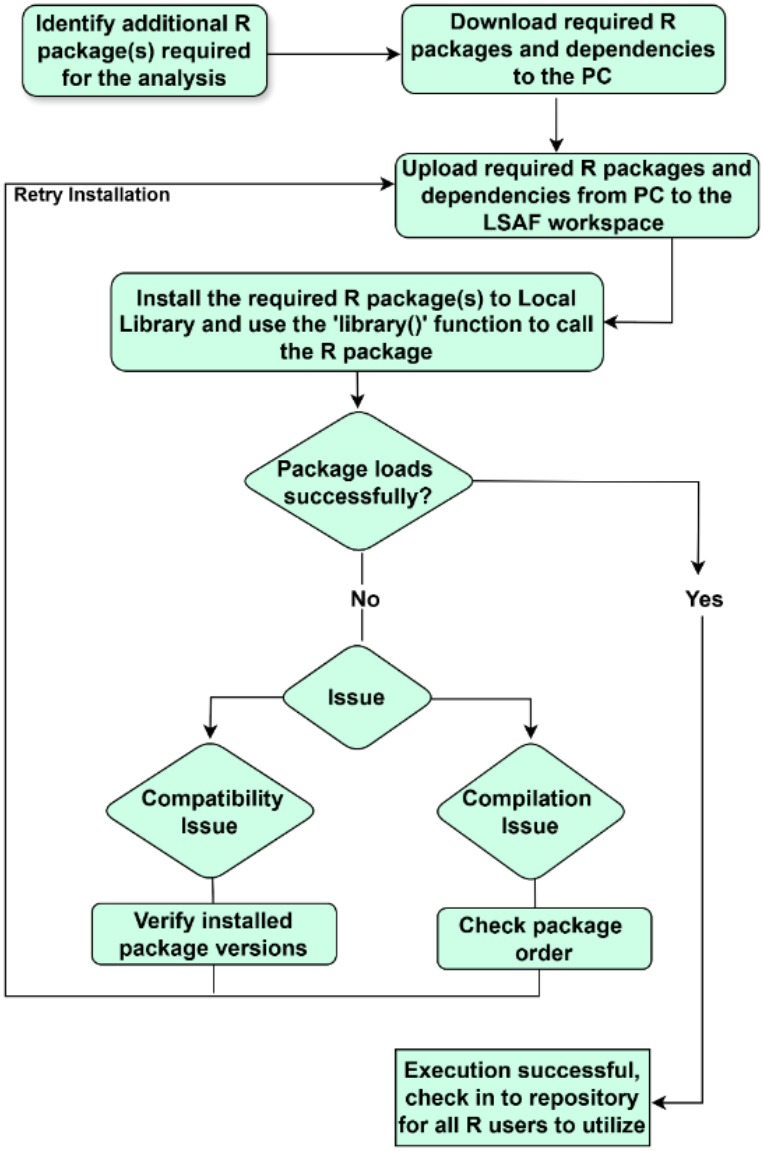
```
CRAN_local <- paste0("file:///",LSAF_WS, "/CRAN")
localLib <- paste0(LSAF_WS, "/R/x86_64-redhat-linux-gnu-library/4.4")
install.packages("r2rtf",repos=CRAN_local,lib=localLib,type="source")
remove.packages("r2rtf", lib = localLib)
library("r2rtf")
```

Users can also create a local CRAN-like repository using the **miniCRAN** R package within RStudio on their PC. This local repository can then be uploaded to LSAF, allowing R packages to be installed directly into the designated R library.

Use miniCRAN to create a local CRAN repository on the user's PC. The makeRepo() function in miniCRAN creates a repository.

```
#Install the 'miniCRAN' package
install.packages("miniCRAN")
# Load the 'miniCRAN'
library(miniCRAN)
# Step 1: Set CRAN mirror
cran_mirror <- "http://cran.rstudio.com"
# Step 2: Specify packages you want to include
pkgs <- c("pharmaversesdtm")
# Step 3: Create directory for local repo (change this path as needed)
local_repo <- file.path(tempdir(), "miniCRAN")
dir.create(local_repo, showWarnings = FALSE)
# Step 4: Get package dependencies recursively
pkg_deps <- pkgDep(pkgs, repos = cran_mirror, type = "source",
                                suggests = FALSE)
print(pkg_deps)  # List of packages including dependencies
# Step 5: Download packages and dependencies to local repo
makeRepo(pkg_deps, path = local_repo, repos = cran_mirror, type =
c("source", "win.binary"))
```





- User identifies additional R package(s) required for the analysis.
- User downloads the required R packages and dependencies to the PC's local or network drive.
- User uploads required R packages and dependencies from the PC to the LSAF workspace.
- User installs the required R package(s) in the local library and calls the R package using the 'library()' function.
- User tests that the package(s) execute successfully and checks them into the repository for all R users to utilize.

Figure 2. Flow diagram of R Package Management in LSAF

CONCLUSION

- LSAF version 5.3 and later integrates R, enabling users to develop and execute programs for clinical data analysis within the platform.
- The LSAF environment includes a standard set of pre-installed R packages and does not allow users to directly download additional packages from CRAN or other external sources.
- If additional packages are required for analysis, users can install them by creating a local library or setting up a CRAN-like repository within their LSAF workspace or project repository.
- LSAF operates on the Linux operating system (Red Hat Enterprise Linux), where R packages are typically installed from source archives (.tar.gz), as binary installations are generally not supported.
- LSAF currently supports R version 4.4.0 (Puppy Cup). Keeping the R version in LSAF up to date would help resolve compatibility issues with the latest R packages.

REFERENCES

SAS Institute Inc. 2021. SAS® Life Science Analytics Framework 5.4: User's Guide. Cary, NC: SAS Institute Inc.

Becker, Matt, and Pritesh Desai. 2025. "Leveraging R for Statistical Analysis in LSAF Q&A, Slides, and On-Demand Recording." *SAS Ask the Expert Webinar*, January 13, 2025. Available at: https://communities.sas.com/t5/Ask-the-Expert/Leveraging-R-for-Statistical-Analysis-in-LSAF-Q-A-Slides-and/ta-p/955967

ACKNOWLEDGMENTS

I want to thank my managers, Kevin Najarian and Joe Luckanish, at Philips and ICON plc, for their guidance and support during the preparation of this paper.

CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Contact the authors at:

Author Name: Praneeth Adidela
Company: ICON plc
Email: praneeth.adidela@iconplc.com
praneethadidela@gmail.com

Author Name: Sagar Koona
Company: ACL Digital
Email: sagar.koona@acldigital.com