Optimizing R Add On Package Management: Seamless Management from Download to Deployment

Vikrant Singh Bisht, Sanofi, Hyderabad, India
Lalit Mehta, Inventive Mind Consultancy Services, India

HYDERABAD, INDIA
PharmaSUG SDE
APRIL 12, 2025

Abstract

R, an open-source programming language, offers a vast array of dedicated packages tailored to specific functionalities. The pharmaceutical industry relies heavily on dedicated functions contained in these packages for Regulatory Submission. This poster aims to unravel the power of R for streamlining package management right from download and installation to test loading R packages, ensuring a qualified environment with robust functionality and compatibility, facilitating thorough package tracking and management to enhance accessibility and reliability for pharmaceutical research and development.

Introduction

With a plethora of add-on packages at their disposal, programmers have a wealth of choice when selecting packages. Often, multiple packages are available for the same functionality, each package with its own pros and cons. To make an informed decision, one must consider various parameters in line with the organization's norms for open-source software. A qualified and versioned environment goes a long way in meeting the usual expectations of quality, traceability and transparency.

R PACKAGE TYPES

- **BASE**
  These packages are available by default in an R instance. They provide essential functions and tools for data analysis, statistical computing, and graphical representation. Approximately 30 packages come bundled with a basic R installation.

- **ADD ON**
  These packages enhance base R functionality with additional tools and methods for data analysis, statistical computing, and graphics. Created by the R community, they can be installed from repositories like CRAN, Bioconductor, RForge, and GitHub.

Key steps in Add On Package Deployment

When faced with Business requests to deploy Add On packages in an existing R environment while adhering to the standard qualification steps, the R system administrator might face a stiff task to ensure that no aspect of qualification is missed.
Qualification is particularly challenging when the R environment is shared by multiple users who continuously request installation of new packages or upgrades to the existing ones, thereby inundating the R system administrator with multiple requests.
In their day-to-day work, an R administrator might most likely use a workflow similar to the one shown here for handling Business requests related to installation of R Add On packages.

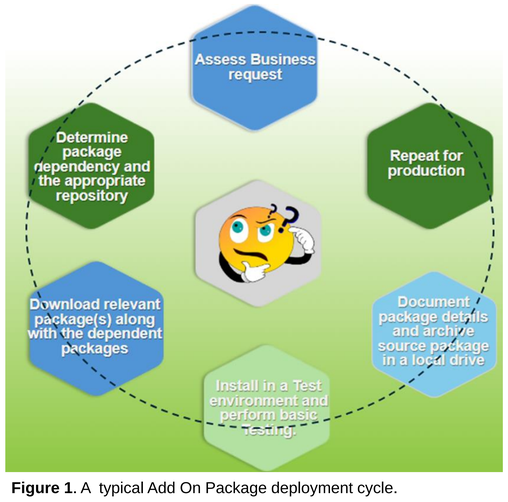


Figure 1. A typical Add On Package deployment cycle.

Workflow for seamless Add On package deployment

The Add On package deployment process can be automated to reduce the burden of manual deployment on the administrator.

1. R administrator will simply trigger an R script. The system takes care of package download, installation and basic validation and also resolves dependencies
2. The system will output a report detailing helpful information related to the packages.

An important point to note is that Add On package versions available on the standard repositories may change at a relatively quicker pace than what developers can manage. Therefore, to maintain traceability, the system also provides the provision to archive all packages that are installed (along with all dependent packages).

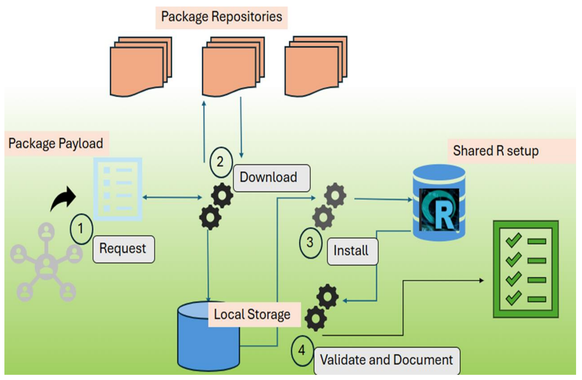


Figure 2. Proposed flow for package deployment

1. ***Request*** – The package(s) requested by end user(s) are listed in a csv payload file.

```
## Read the package list from a standard csv file on the
server ##
packages <-
readLines("/path/to/packagelist/package_list.csv")
```

2. ***Download*** – The automation reaches out to the package repository for downloading the package(s) along with dependencies. The code loops over the character vector created earlier to process each package and download it.

NOTE – Since the installation will be done using `download.packages`, it is essential to index the downloaded packages using the function `write_PACKAGES`. Indexing is essential since it generates the `PACKAGES` and `PACKAGES.gz` index files, which contain metadata about the packages in the directory. These files serve as a map for the subsequent installation step to track package dependencies (any other package on which the package depends or any package which it imports etc.)

```
##Download and Index the package list##
download.packages(packages,destdir="/path/to/packagerepo/"
, type="source")
write_PACKAGES("/path/to/packagerepo/")
```

3. ***Install*** – Once packages have been downloaded and archived locally, they will be installed one by one. Only packages that were downloaded earlier will be installed.

```
## Prepare the package list from download location. Change
setwd path for modifying location ##
setwd("/path/to/packagerepo/")
install_packages <- system("ls -l | grep .tar.gz |awk
'{print $9}' | cut -d _ -f 1",intern=TRUE)
writeLines(install_packages, "downloaded.csv")

## Install the package list from download location. Change
setwd path for modifying location ##
installedpkgs <-
getPackagelist(readLines("downloaded.csv"))
install.packages(installedpkgs,contriburl=(contrib.url="fi
le:///path/to/packagerepo/"))
```

4. ***Validate and Document*** - Perform basic load testing on each package and prepare a final consolidated list with package details

```
## Validate Installed Packages ##
cat("Library paths being checked: \n")
print(.libPaths())
#Get list of installed packages across all libraries
installed_packages <- installed.packages(lib.loc =
.libPaths())
inst_pack <- installed_packages
package_names <- inst_pack[,"Package"]
## Consolidate & document Packages details. ##
write.csv(installed.packages(), "installed.csv")
```

Licensing Considerations

An important aspect to consider prior to deploying Add On packages within a qualified R environment is licensing. With critical clinical data at stake, it is of utmost importance to ensure that licensing commitments are clearly understood and documented. This might also be in line with organizational expectations The different type of licenses are listed in table shown in Figure 3.

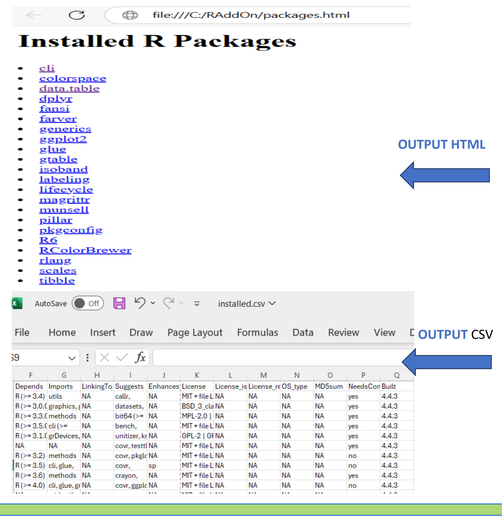
License Type	Description	Usage
GNU General Public License (GPL)	Freely use, modify, and distribute the code; derivative work must also be under GPL.	Commonly used for open-source projects.
MIT License	Very permissive; use, modify, and distribute with minimal restrictions.	Popular for its simplicity and permissiveness.
Apache License 2.0	Permissive; includes patent protection and requirements to note changes.	Often used for projects requiring patent protection.
BSD License	Permissive with minimal restrictions; used in academia.	Common in academic and research settings.
Creative Commons Licenses	Used for data packages; CC0 allows free use, CC BY requires attribution.	Used for data packages and datasets.

Figure 3. Licensing Details

SAMPLE CODE

```
## This code will download, install, validate &
create an output html with clickable links. The
code is optimized for Windows R and updates are
needed for running in a Linux environment. Non CRAN
packages are outside the scope of this code.##
##NOTE - Please ensure before running this code:##

##1. Package data.table and its dependencies have
been installed manually.##
##2. Output directory has been created.##
##3. Payload file payload.csv is pre populated with
the packages is placed in the input folder. ##

## Load required packages ##
library("data.table")
library("tools")

## Setup an array for BASE R Packages ##
arr_base_packs =
c(rownames(installed.packages(priority =
c("base","recommended"))),"translations")

## Setup the repository configuration##
local({r <- getOption("repos");
r["CRAN"] <- "http://cran.r-project.org";
options(repos=r)})

## This function creates a Dependencies and Imports
List. ##
fetchPkgs <- function(pkgs){
packages <-unlist(tools::package_dependencies(pkgs,
available.packages(),

## If Suggests is needed, add to the comma
separated list. ##
which=c("Depends", "Imports"), recursive=TRUE)
  )
  packages <- union(pkgs, packages)
  packages <-data.table(packages,key=c("packages"))

## Ensure that Base packages are not overwritten ##
data_frame_base_packs <-
data.table(arr_base_packs,key=c("arr_base_packs"))
packages <- packages[!data_frame_base_packs]
packages <- packages$packages
}

## Read package list from a standard csv file ##
packages <-fetchPkgs(readLines("C:\\payload.csv"))


## Download and Index the package list. Set type =
"source" for Linux.##
download.packages(packages,destdir="C:\\RAddOn\\Dow
nloads",type="win.binary")
write.csv(list.files("C:\\RAddOn\\Downloads"),"C:\\
RAddOn\\download.csv")
write_PACKAGES("C:\\RAddOn\\Downloads",
type="win.binary")

## Prepare package list from download location.
Change setwd path for modifying location##
setwd("C:\\RAddOn\\Downloads")
download_packages <-
sapply(strsplit(list.files(pattern="\\.zip"),'_'),f
unction(split) split[1])
writeLines(download_packages,"C:\\RAddOn\\downloade
d.csv")

## Comment the previous 2 lines and uncomment the
following 2 lines for Linux R instance ##
#install_packages <- system("ls -l | grep
.tar.gz|awk '{print $9}' | cut -d _ -f
1",intern=TRUE)

#download_packages <-
sapply(strsplit(list.files(pattern="\\.tar.gz"),'_'
),function(split) split[1])

## Install packages from download location. ##
installedpkgs <-
fetchPkgs(readLines("C:\\RAddOn\\downloaded.csv"))
install.packages(installedpkgs, contriburl =
(contrib.url="file:///C:\\RAddOn\\Downloads\\"),ty
pe="win.binary")

## Test Load installed packages ##
cat("Checking installed packages in all the
Library Paths: \n")
print(.libPaths())

## Get installed packages across libraries ##
installed_packages <- installed.packages(lib.loc =
.libPaths())
inst_pack <- installed_packages
package_names <- inst_pack[,"Package"]
for (pack in package_names)
{
 tryCatch({

   library(pack,character.only = TRUE)
   cat("Successful load for package:",pack,"\n")
         },error=function(e)
         {
          cat("Failed to load package:
",pack,"Error:",e$message,"\n")
         })
}

## Setting path to point to the desired output
location for HTML ##
setwd("C:\\RAddOn")

## Consolidate FINAL Installed Package List ##
write.csv(installed.packages(), "installed.csv")

## Output HTML for installed packages ##
final_installed <- rownames(installed.packages())
html_content <- "<html><head><title>Install
DefineDocument</title></head><body><h1>Installed R
Packages</h1><ul>"
for (pkg in final_installed)
{
html_content <- paste0(html_content, "<li><a
href='https://cran.r-
project.org/package=",pkg,"'>",pkg,"</a></li>")
}
html_content <- paste0(html_content,
"<ul></body></html>")
writeLines(html_content,"packages.html")
```

SAMPLE OUTPUT



Contact Information

Your comments and questions are valued and encouraged. Contact the author at:

Name: Vikrant Singh Bisht
Company: Sanofi India
E-mail: vikrantz887@gmail.com
Web: https://www.linkedin.com/in/vikrant-bisht-433451229/

The ideas/analysis shared here belong solely to the individual and not necessarily shared by the employer.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.