PharmaSUG China 2023 - Paper CC-157

R Package Installation Made Easy

Jie Wang, Janssen (China) Research & Development;

ABSTRACT

More and more companies and organizations in our pharmaceutical industry are adopting R to their statistical programming process, using it for data analysis, reporting and visualization. However, sometimes, installing and managing R packages can be challenging for beginners and experts alike. This paper provides some more details on package structure and state (source package, bundled package, binary package, installed package etc.), and different methods to install R packages from various sources, such as CRAN, GitHub, and local files. And how to solve some common R package installation errors. Will also introduce some more tools on how to check package dependencies and practical tips on working with R `renv` package to install and manage your RStudio Project Environments.

INTRODUCTION

In general, installing R packages can be quite easy and most R packages can be installed directly from the R console using the `install.packages()` function. Some packages may require additional dependencies or libraries, which may need to be installed separately. Additionally, some packages may have specific system requirements, such as a certain operating system or processor architecture.

Two common issues I encountered when I was working on two different projects on our company SCE (Statistical Computing Environment):

- For {Figure 1: conquer} package: this is an indirect dependency that the project needs. But due to the fact that the SCE only installed old version of `gcc` compiler, this package which used `C++17`, was failed to compile.

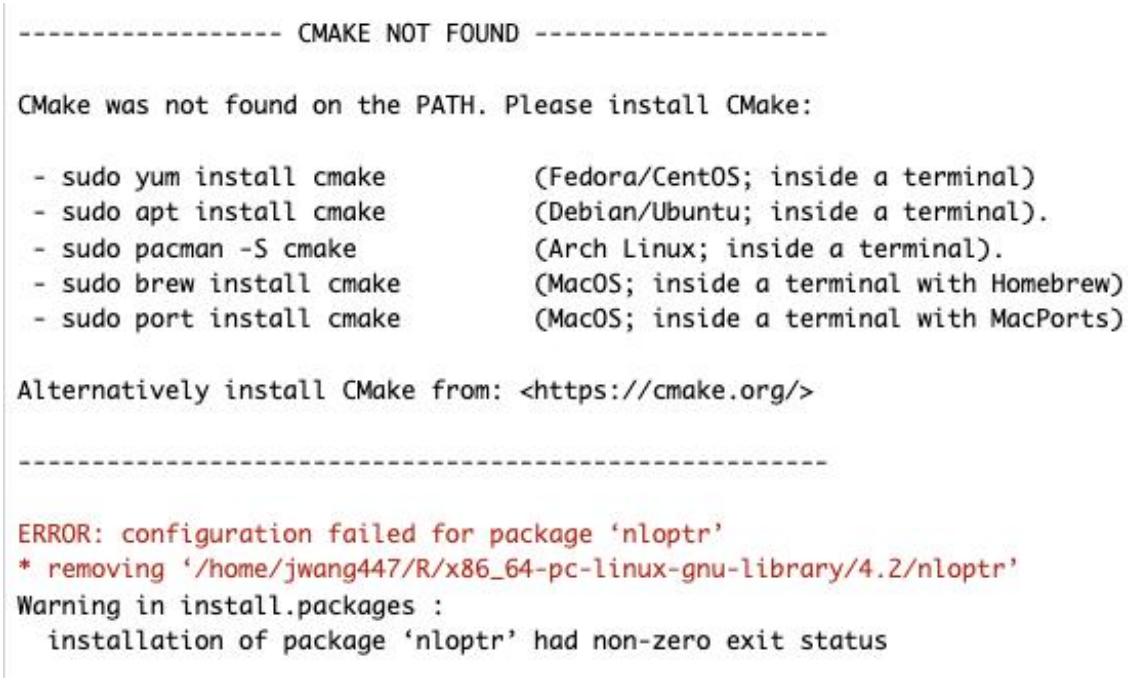
Figure 1: conquer

```
> install.packages('conquer')
Installing package into '/home/jwang447/R/x86_64-pc-linux-gnu-library/4.2'
(as 'lib' is unspecified)
trying URL 'https://cloud.r-project.org/src/contrib/conquer_1.3.3.tar.gz'
Content type 'application/x-gzip' length 56315 bytes (54 KB)
==================================================
downloaded 54 KB

* installing *source* package 'conquer' ...
** package 'conquer' successfully unpacked and MD5 sums checked
** using staged installation
** libs
Error: C++17 standard requested but CXX17 is not defined
* removing '/home/jwang447/R/x86_64-pc-linux-gnu-library/4.2/conquer'
Warning in install.packages :
  installation of package 'conquer' had non-zero exit status
```

- For {Figure 2: nloptr} package: : this was also an indirect dependency needed for the ”MMRM” analysis. But it depends on ‘cmake’ which the SCE system does not include, so the installation failed.

Figure 2: nloptr



R PACKAGE STATE AND STRUCTURE

Before I introduced the workarounds that I tried to fix both problems as shown above, I think it is beneficial to understand the R package state and structure[1].

Package State	Notes (Example)
Source package	a directory of files with a specific structure (Figure 3: {fs} source package).
Bundled package	a package that has been compressed into a single platform-agnostic ‘tarballs’, with extension “.tar.gz” (Figure 4: {fs} bundled package)
Binary package	platform specific single file with extension:

1

Package State	Notes (Example)
	- MacOS: .tgz - Windows: .zip - Linux: tools needed to install from bundled package
Installed package	a binary package that has been decompressed into a package library (Figure 5: {fs} binary package)
In-memory package	an installed package that has been loaded into memory

Figure 3: {fs} source package

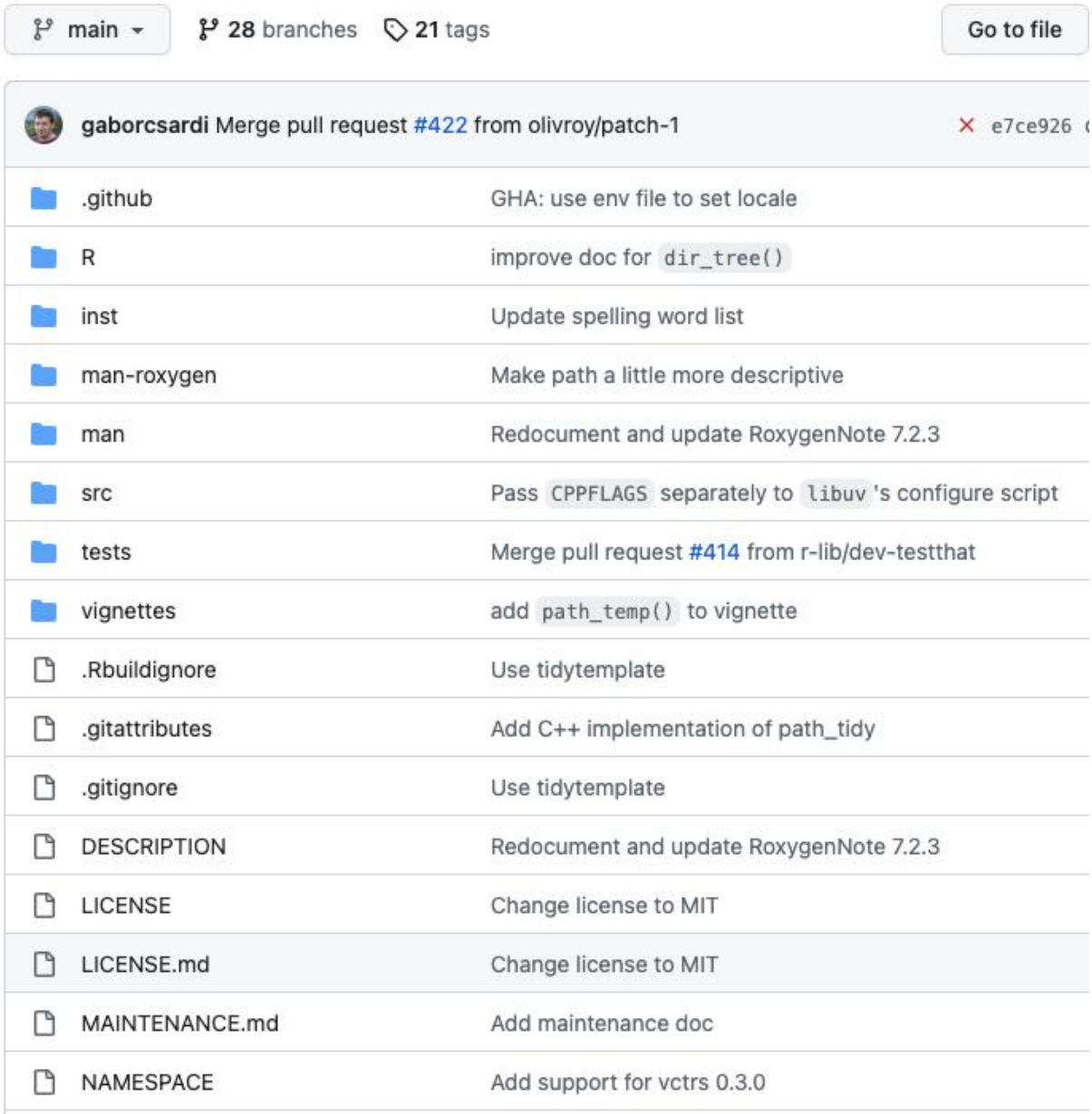


Figure 4: {fs} bundled package: folder tree structure

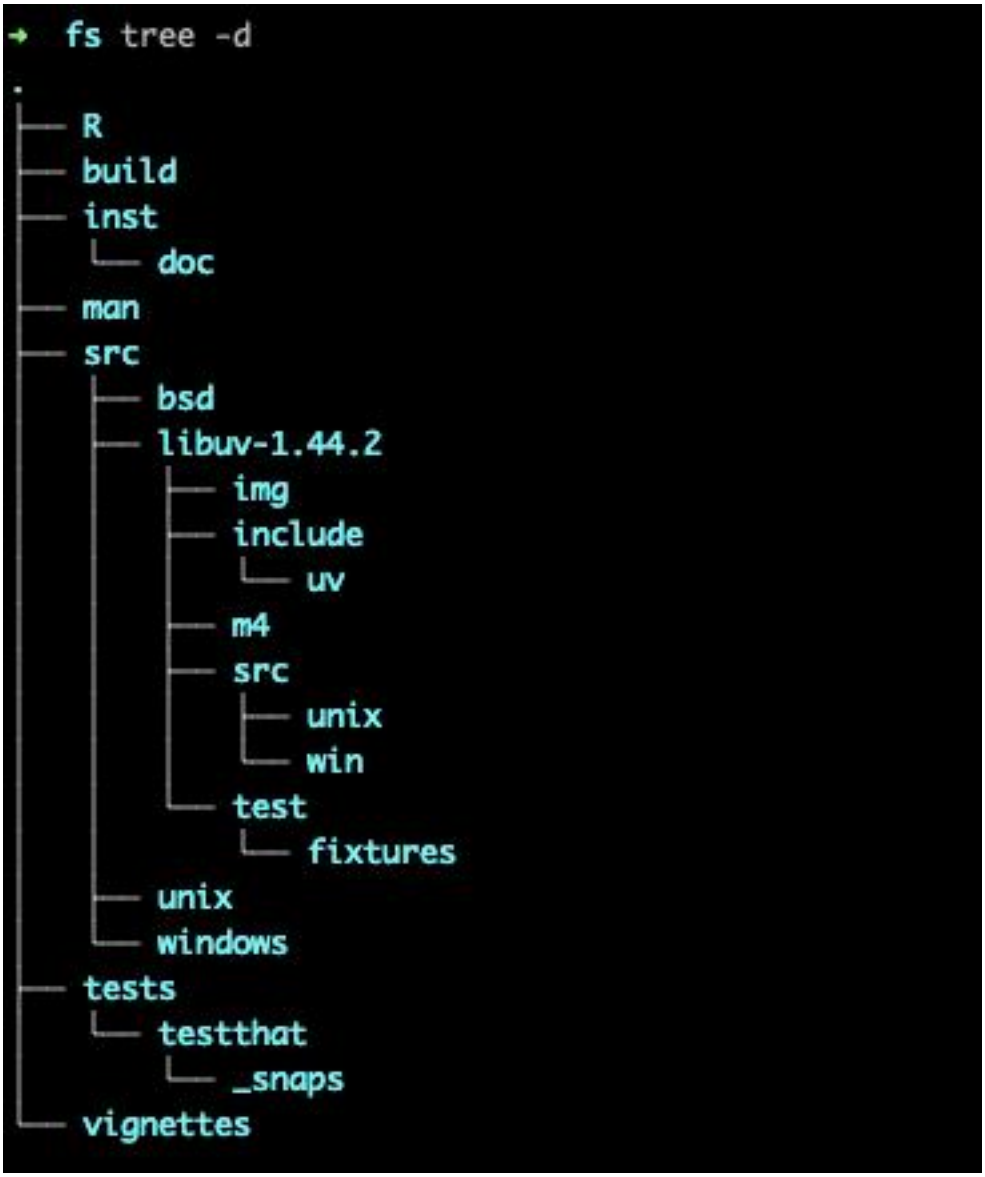


Figure 5: {fs} binary package: binary for Windows

INSTALL R PACKAGE FROM VARIOUS SOURCES

This section will discuss about the most commonly used commands for installing R packages from various sources and also discuss about a promising alternative {pak}.

MOST COMMONLY USED COMMANDS

Command	Note
Install.packages("fs")	Install package {fs} from CRAN
devtools::install_github("r-lib/fs") remotes::install_github("r-lib/fs")	Install from GitHub; {devtools} re-export functions from the {remotes}
install.package("path/to/fs", repos=NULL, method="source") devtools::install_local("fs")	Install from local
Devtools::install_version("fs", "1.6.0") renv::install("fs@1.6.0")	Install a specific version of {fs}

PROMISING ALTERNATIVE {PAK}

This paragraph uses the PaperBody style.

Command	Note
pak::pkg_install("fs")	Install package {fs} from CRAN
pak::pkg_install("r-lib/fs")	Install package {fs} from GitHub
pak::pkg_install("r-lib/fs@1.5.0")	Install a specific version or tag of {fs} from GitHub
pak::local_install("fs")	Install {fs} from a local path

R PACKAGE INSTALLATION RELATED

This section will discuss some R package installation related options and tools:

CRAN MIRROR

CRAN (Comprehensive R Archive Network) is the official repository for R packages. The CRAN mirrors are a distributed network of servers located worldwide. You could find all available mirrors here (CRAN - Mirrors (r-project.org)) and choose the mirror nearest to your current location to maximize the R package download speed.

Command	Note
options("repos") getOption("repos")	Check what is the current CRAN mirror used
options(repos="xx")	Set the CRAN mirror e.g. options(repos ="https://cloud.r-project.org") options(repos ="https://mirrors.sjtug.sjtu.edu.cn/cran/")

ENVIRONMENT VARIABLES – R_LIBS_USER / R_LIBS_SITE

Both R_LIBS_USER abd R_LIBS_SITE can change the value of search paths for R packages. Users can have one or more libraries, normally specified by the environment variable R_LIBS_USER and R_LIBS_SITE. For the R_LIBS_USER, this has a default value (to see it, use 'Sys.getenv("R_LIBS_USER")' within an R session), but that is only used if the corresponding directory actually exists (which by default it will not). Sys.getenv("R_LIBS_SITE") will give the locations of the site libraries if available.

R .RENVIRON AND .RPROFILE

.Renviron file is most useful for defining sensitive information such as API keys (such as GitHub) as well as R specific environment variables like default library locations R_LIBS_USER as mentioned above.

The .Rprofile file contains R code to be run when R starts up. It is run after the .Renviron file is sourced.

TOOLS

NAME	Note
Posit Package Manager	Posit Public Package Manager, contains binary packages for all major platforms including Linux
Rtools for Windows	Rtools is a toolchain bundle used for building R packages from source for Windows users
R-universe	Personal package repositories for R
rig	Cross-platform R Installation Manager
R package {pak}	R package installation and package dependency checking

CONCLUSION

Installing R packages is crucial to harnessing the full power and functionality of R and often the first step for modern data analysis. Hope those details and tools around R package installation will help you out when you encounter similar issues. Happy R Coding.

REFERENCES

R Packages (2e) – Hadley Wickham and Jennifer Bryan (R Packages (2e) (r-pkgs.org))

What They Forgot to Teach You About R (rstats.wtf)

CONTACT INFORMATION

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

Jie Wang
Janssen (China) R & D
JWang447@its.jnj.com

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