# Simple process for sharing R code via Docker

Ronald (Ryy) Glenn Thomas

2024-07-21

## Table of contents



## 1 Introduction

Its often the case that two research collaborators want to work on the same `R` codebase. Unfortunately, to accomplish this, its often not as simple as one researcher sending a text file containing the code to the other colleague.

For example, a number of elements of the computing environment can differ between collaborators, such as: the version of `R`, the operating system(macos vs. windows vs Linus vs Other), `R` packages installed, the versions of the packages, startup files

(e.g. `.Rprofile`) and environment variables. Any of these can cause code working for one reseracher to break when run by a second.

## 2 Methods

Lets assume you have some `R` program code in a file, say `peng.R`, that you're written to analyze some "Palmer Penguin" data. You want to share the code with a colleague, we'll call him Joe. How to proceed?

The simplest option is simply to send Joe the `R` file containing the code via the most convenient method (e.g. email)

The next step is for for Joe to (attempt to) run the code with the idea of expanding the analysis. Typically he would do this either using an IDE, such as `Rstudio`, or run the code from the command line in the terminal with the command:

```
> R -e "source('peng1.Rmd')"
```

Sometimes this approach works. When it does, Joe can add comments or expand the code and relay it back to you, "rince and repeat" and all is well. Frequently, however, this simple approach will fail for any of several reasons. Even when it runs, unless care is taken, its not guaranteed that Joe will get the same results you do.

Ideally to facilitate reproducibility your colleague Joe will have a computing environment as similar to yours as possible. This can be difficult to achieve, especially given the dynamic nature of open source software. For example, Joe may have an older version of `R` installed on his workstation, or his `R` environment may be missing a necessary package to run the code or it may be the wrong version. Additiional potential problems include: the program may need to source an additional file or load data that's missing.

All of these problems go away if instead of sending the program as a standalone text file you send it as a docker image. In this post we'll walk through the process of dockerizing the R code.

## 3 Methods

Assume we have a simple `R` file that we want to share with Joe such as the following:

````
---
title: "\\includegraphics[width=2cm, right]{sudoku_apple.pdf}\\newline
  New Penguin plot"
author: "R.G. Thomas"
date: "`r format(Sys.time(), '%B %d, %Y')`"
fontsize: 11pt
geometry: "left=3cm,right=5cm,top=2cm,bottom=2cm"
output:
  pdf_document:
    keep_tex: true
    includes:
header-includes:
   - \usepackage{lipsum}
   - \usepackage[export]{adjustbox}
   - \usepackage{currfile}
   - \usepackage{fancyhdr}
   - \pagestyle{fancy}
   - \setlength{\headheight}{14pt}
   - \fancyfoot[L]{\currfilename} %put date in header
   - \fancyfoot[R]{\includegraphics[width=.8cm]{sudoku_apple.pdf}}
   - \fancyhead[L]{\today} %put current file in footer
   - \fancyhead[R]{Penguins RGT Lab report}
latex_engine: xelatex
---

```{r  echo=F}
# clear env: objects and packages
library(pacman)
p_load(palmerpenguins, rmarkdown,  knitr )

opts_chunk$set(
  warning = FALSE, message = FALSE, echo = FALSE, fig.width = 5.2,
  fig.height = 3, results = "asis", dev = "pdf"
)
```
````

```
# Introduction

\lipsum[1-5]

Draft report examining Penguin characteristics. Data from Alison Horst article
in the `R journal`
@m.horstPalmerArchipelagoPenguins2022

```{r }
attach(penguins)
plot(flipper_length_mm, bill_length_mm, col=sex)
legend("topleft", legend=levels(sex), pch=16, col=unique(sex))
```

\lipsum[1-5]
\bibliography{penguins.bib}
\bibliographystyle{plain}
\nocite{*}
```

## 4 Share program code with Joe.

Joe downloads the attachment. Opens a working directory and attempts to run the Rmd file

with the command

```
> R -e "source('peng.R')"
```

Joe has a linux mint desktop

```
> mkdir peng_collaboration
> cd peng_collaboration
> R -e "source('peng.R')"
```

Linux can't find R

Joe can fix this by installing `R`

```
> sudo apt install r-base-core
```

## 5 Docker approach

Alternatively, consider the "Docker" approach.

Before sending peng.Rmd to Joe we'll dockerize it.

- Prepare a work directory: peng1. We want to prepare a repo and a docker image that has a standardized computing environment set up including R, the R packages and all the preliminaries taken care of, so that all Joe has to do is clone the repo and run the image from dockerhub.

Steps for Joe:

- install docker on his machine.
- download docker image from dockerhub
- run image

Here is the docker file

```
FROM rocker/r-devel
# FROM rhub/r-minimal
RUN apt-get install pandoc -y
# RUN R -e "install.packages(c('rmarkdown','palmerpenguins'), repo='cran.rstudio.com')"
RUN R -e "install.packages('renv', repos = c(CRAN = 'https://cloud.r-project.org'))"

RUN groupadd --system joe
RUN useradd --system --gid joe -m joe
RUN chown joe:joe -R /home/joe
USER joe
WORKDIR /home/joe
COPY renv.lock renv.lock
RUN mkdir -p renv
COPY .Rprofile .Rprofile
COPY renv/activate.R renv/activate.R
COPY renv/settings.json renv/settings.json
RUN R -e "renv::restore()"
RUN mkdir -p /home/joe/shr
```

```
RUN mkdir -p /home/joe/output
CMD ["/bin/bash"]
```

run docker

```
docker build -t rgt47/penguin_review --platform=linux/amd64 .
docker push rgt47/peng_review
```

relay image to Joe

```
docker push rgt47/peng_review
```

or

```
docker save rgt47/peng_review | gzip > peng_review_trans.tgz
docker load -i peng_review_trans.tgz
```

```
> docker pull rgt47/penguin_review

droot="$PWD"/output docker run -it --rm -v $droot:/home/joe/output rgt47/penguin1

droot="$PWD" docker run -it --rm  \
-v $droot/output:/home/joe/output -v $droot/share:/home/joe/share rgt47/penguin1

> cd output
> library(rmarkdown); render('../shr/peng.Rmd')
```

Important to include the association between the /home/joe/output directory in the container with the output directory on the local workstation. Thats where the results of the analysis will be saved.

```
> R -e "library(rmarkdown); render('peng.Rmd')"
```

and if he wants to edit peng.Rmd

```
> vim peng.Rmd
```

## 6 REFERENCES

Running your R script in Docker

```
docker run --rm -p 8787:8787 -e PASSWORD=z rocker/rstudio

j share
cd peng1

docker build -t rgt47/penguin1 .
docker run --rm -it rgt47/penguin1

droot="${PWD}"

docker run -it --rm -v $droot:/home/joe/output rgt47/penguin1

Problem
  docker run -it --rm -v $droot:/home/joe rgt47/penguins1

Unable to find image 'rgt47/penguins1:latest' locally
docker: Error response from daemon: manifest for rgt47/penguins1:latest not found: manifest unk
See 'docker run --help'.
share_R_code_via_docker/peng1   echo $droot
/Users/zenn/prj/qblog/posts/share_R_code_via_docker/peng1
share_R_code_via_docker/peng1   mkdir output
share_R_code_via_docker/peng1   f

Need a base R that includes dev tools

Try rocker/r-devel

docker build -t rgt47/penguin2 . --platform linux/amd64


docker run --rm \
  -e DISABLE_AUTH=true -e ROOT=true \
  -v "$(pwd):/home/rstudio/project:rw" \
```

```
  -p 8787:8787 \
  --name rstudio rocker/verse:3.6.3

docker run -it --rm -v "$(pwd):/home/joe" rgt47/penguins1
```