Using multiple R versions on Linux

Alex M. Chubaty

29 Jun 2023

Contents

Overview

Unlike on Windows, most R users on linux tend to work with a single version of R – whichever version is available in their software repositories. With a little bit of work, it is possible to use multiple versions of R on Linux; however, you will need an account with `sudo` privileges to set this up.

The most common use case is to install R-devel alongside the current R version in order to facilitate package checking and testing prior to CRAN submission. Additionally, you may wish to maintain older versions of R to maintain compatibility and reproducibility of old projects, scripts, etc.

This document provides an overview of several approaches to working with multiple versions of R on Linux - although these can also be used on macOS and Windows.

1. Using Docker

This is the preferred solution, as it is self-contained. Every time you start an instance of R in a Docker container, it starts in a "factory fresh" state, which means you will need to install additional packages into each new instance or save (commit) your changes for later reuse. The advantages here are that packages installed for each project are kept separate, ensuring proper reproducibility and portability. (Although `renv` is also good for maintaining separate package libraries per project, it doesn't ensure system dependency separation.)

Please note that this guide is intended only to get you started with using R with Docker, not to be a full tutorial for more advanced Docker use. See the official Docker documentation and the rocker project for more advanced usage, including using containers with RStudio Server installed[1] or using a virtual X server.

Older R versions

1. Install Docker following the instructions here.
2. Download the image for the version of R you wish to use:

   ```
   docker pull r-base:3.6.3
   ```

3. Start a container running the R version of your choice:

   ```
   docker run -it --rm r-base:3.6.3
   ```

   * NOTE: the `--rm` flag tells docker to remove the container after use. If you wish to keep the container for reuse later (*e.g.*, so you don't need to reinstall packages, etc.), then omit this.

R-devel

1. Install Docker following the instructions here.
2. Download the latest R-devel image:

   ```
   docker pull rocker/drd
   ```

3. Start a container running R-devel:

   ```
   docker run -it --rm rocker/drd Rdevel
   ```

   * NOTE: the `--rm` flag tells docker to remove the container after use. If you wish to keep the container for reuse later (*e.g.*, so you don't need to reinstall packages, etc.), then omit this.

Using RStudio

Alternatively, there are docker images that include RStudio server built-in, so you can run a container and connect to it through your browser instead of having to work in a terminal.

1. *E.g.*, start a versioned container running RStudio that includes geospatial libraries from the unstable ubuntugis repo:

   ```
   docker run --rm -ti -p 127.0.0.1:8080:8787 rocker/geospatial:4.2.3-ubuntugis
   ```

2. Look for the user password to appear (in red) once it's done setting up the container.
3. Open a web browser and go to `localhost:8080` to get to the RStudio instance running in the docker container.
4. Log in with username `rstudio` and the password from step 2.

[1] https://github.com/rocker-org/rocker-versioned2

2. Using `rig`

Using the R installation manager `rig` (https://github.com/r-lib/rig) is the easiest approach, as it automates the installation an linking of different R versions, and is cross-platfrom (works with Windows, macOS, and Linux). Although each version of R maintains separate package libraries, users should strive to use per-project package libraries, e.g., using `renv`.

1. Install `rig` using the platform-specific installation instructions are provide at https://github.com/r-lib/rig#id-installation;
2. Install the R versions you need:

   ```
   rig add release
   rig add devel
   ```

   ```
   rig add 4.1
   rig add 4.2
   rig add 4.3
   ```

   You can see the available versions (and the default, starred) by using `rig list`.

   Set the default version using the names from `rig list`:

   ```
   rig default 4.3-arm64
   ```

3. Setup symlinks to allow easy launch of versioned R sessions:

```
rig system make-links
```

Use e.g., `R-4.3-arm64` on macOS.

4. On macOS, restrict access to system package directories:

   ```
   rig system fix-permissions
   ```

5. On macOS can also use the menu bar app (https://github.com/r-lib/rig#id-macos-menu-bar-app) to manage their R versions:

   ```
   open -a Rig
   ```

Be sure to open the Rig app preferences to allow launch at startup.

Switching R versions using `rig`

Setting the default version

```
## e.g., on macOS:
rig default 4.3-arm64
```

Users on macOS can also use the menu bar app.

Selectively launching an RStudio seesion using specific version of R

```
## e.g., on macOS:
rig rstudio 4.2-arm64
```

Users on macOS can also use the menu bar app.

3. Manual build + installation

As noted above, using Docker or `rig` are likely better solutions, as manual installations require manual upkeep and manual *uninstallation*.

Older R versions

1. Install prerequisites for building R from source:

   ```
   sudo apt build-dep r-base
   ```

2. Create directories to keep the R source code:

   ```
   mkdir ~/R/src
   ```

3. Get the source code for the version of R you wish to install:

   R 3.y.z:

   ```
   cd ~/R/src/
   RVERSION=3.6.3
   wget https://cran.r-project.org/src/base/R-3/R-${RVERSION}.tar.gz
   tar -xvzf R-${RVERSION}.tar.gz
   ```

   R 4.y.z:

   ```
   cd ~/R/src/
   RVERSION=4.2.3
   wget https://cran.r-project.org/src/base/R-4/R-${RVERSION}.tar.gz
   tar -xvzf R-${RVERSION}.tar.gz
   ```

4. Install:

   ```
   cd ~/R/src/R-${RVERSION}
   ./configure --prefix=/usr/local/lib/R-${RVERSION} --enable-R-shlib --with-blas --with-lapack
   make
   sudo make install
   ```

5. Create symlink:

   ```
   sudo ln -s /usr/local/lib/R-${RVERSION}/bin/R /usr/local/bin/R-${RVERSION}
   ```

6. Run a specific R version:

   From the commandline, simply do (according to `${RVERSION}`):

   ```
   R-3.6.3
   # or
   R-4.2.3
   ```

R-devel

1. Install prerequisites for building R from source:

```
sudo apt build-dep r-base
sudo apt install ccache subversion xorg-dev
```

2. Create directories to keep the R-devel source code and to `make` it:

```
mkdir -p ~/R
mkdir -p ~/GitHub/r-config/R-devel/src/
ln -s ~/GitHub/r-config/R-devel/src ~/R/src
```

3. Get the latest version of R-devel from the subversion repository:

```
cd ~/R/src
svn co https://svn.r-project.org/R/trunk r-devel/R
```

4. Link the recommended packages for building R:

```
cd ~/R/src/r-devel/R
bash ./tools/rsync-recommended
bash ./tools/link-recommended
```

5. Use the installation script in `R-devel/scripts/build-R-devel.sh`:

```
## ~/GitHub/r-config/R-devel/scripts/build-R-devel.sh
#!/bin/sh
cd ~/R/src/R-devel-build

# R_PAPERSIZE=letter                          \
# R_BATCHSAVE="--no-save --no-restore"        \
# R_BROWSER=xdg-open                          \
# PAGER=/usr/bin/pager                        \
# PERL=/usr/bin/perl                          \
# R_UNZIPCMD=/usr/bin/unzip                   \
# R_ZIPCMD=/usr/bin/zip                       \
# R_PRINTCMD=/usr/bin/lpr                     \
# LIBnn=lib                                   \
# AWK=/usr/bin/awk                            \
# CC="ccache gcc"                             \
# CFLAGS="-ggdb -pipe -std=gnu99 -Wall -pedantic -DTESTING_WRITE_BARRIER" \
# CXX="ccache g++"                            \
# CXXFLAGS="-ggdb -std=c++0x -pipe -Wall -pedantic" \
# FC="ccache gfortran"                        \
# FCFLAGS="-ggdb -pipe -Wall -pedantic"       \
# F77="ccache gfortran"                       \
# FFLAGS="-ggdb -pipe -Wall -pedantic"        \
# MAKE="make -j4"                             \
# ./configure                                 \
#     --prefix=/usr/local/lib/R-devel         \
#     --enable-R-shlib                        \
#     --enable-strict-barrier                 \
#     --with-blas                             \
#     --with-lapack                           \
#     --with-readline                         \
#     --with-recommended-packages             \
#     --program-suffix=dev
```

```
R_PAPERSIZE=letter                                  \
R_BATCHSAVE="--no-save --no-restore"                \
R_BROWSER=xdg-open                                  \
PAGER=/usr/bin/pager                                \
PERL=/usr/bin/perl                                  \
R_UNZIPCMD=/usr/bin/unzip                           \
R_ZIPCMD=/usr/bin/zip                               \
R_PRINTCMD=/usr/bin/lpr                             \
LIBnn=lib                                           \
AWK=/usr/bin/awk                                    \
CC="ccache gcc"                                     \
CFLAGS="-ggdb -pipe -std=gnu99 -Wall -pedantic"     \
CXX="ccache g++"                                    \
CXXFLAGS="-ggdb -pipe -Wall -pedantic"              \
FC="ccache gfortran"                                \
F77="ccache gfortran"                               \
MAKE="make"                                         \
../r-devel/R/configure                              \
    --prefix=/usr/local/lib/R-devel                 \
    --enable-R-shlib                                \
    --with-blas                                     \
    --with-lapack                                   \
    --with-readline                                 \
    --with-recommended-packages                     \
    --with-tcltk                                    \
    --program-suffix=devel

make

echo "*** Done -- now run 'make install'"
```

6. Give the script execute permissions:

```
chmod a+x ~/GitHub/r-config/R-devel/scripts/build-R-devel.sh
```

7. Make and install R-devel:

```
mkdir -p ~/R/src/R-devel-build
bash ~/GitHub/r-config/R-devel/scripts/build-R-devel.sh

cd ~/R/src/R-devel-build
sudo make install
```

8. Create custom script to launch R-devel in `R-devel/scripts/R-devel.sh`:

```
## ~/GitHub/r-config/R-devel/scripts/R-devel.sh
#!/bin/bash
#export R_LIBS_SITE=${R_LIBS_SITE-'/usr/local/lib/R-devel/lib/R/library:/usr/local/lib/R/site-libra
export R_LIBS_SITE=${R_LIBS_SITE-'/usr/local/lib/R-devel/lib/R/library'}
export PATH="/usr/local/lib/R-devel/bin:$PATH"
R "$@"
```

   Note that this keeps the R-devel package library separate from your regular R libraries.

   Be sure to give the script execute permissions:

```
chmod a+x ~/GitHub/r-config/R-devel/scripts/R-devel.sh
```

9. Create custom script to launch Rscript-devel in `R-devel/scripts/Rscript-devel.sh`:

```
## ~/GitHub/r-config/R-devel/scripts/Rscript-devel.sh
#!/bin/bash
export R_LIBS_SITE=${R_LIBS_SITE-'/usr/local/lib/R-devel/lib/R/library'}
export PATH="/usr/local/lib/R-devel/bin:$PATH"
Rscript "$@"
```

Note that this keeps the R-devel package library separate from your regular R libraries.

Be sure to give the script execute permissions:

```
chmod a+x ~/GitHub/r-config/R-devel/scripts/Rscript-devel.sh
```

10. Create symlinks to launch R-devel and Rscript-devel:

```
sudo ln -s ~/GitHub/r-config/R-devel/scripts/R-devel.sh /usr/local/bin/R-devel
sudo ln -s ~/GitHub/r-config/R-devel/scripts/Rscript-devel.sh /usr/local/bin/Rscript-devel
```

11. Create a script used to update R-devel in `R-devel/scripts/update-R-devel.sh`:

    This could be turned into a `cron` job.

```
## ~/GitHub/r-config/R-devel/scripts/update-R-devel.sh
#!/bin/bash

# update source code
cd ~/R/src/r-devel/R
svn update
bash ./tools/rsync-recommended
bash ./tools/link-recommended

cd ~/GitHub/r-config/R-devel/scripts
bash ./build-R-devel.sh

# make and install
cd ~/GitHub/r-config/R-devel/src/R-devel-build
sudo make install
```

    Be sure to give the script execute permissions:

```
chmod a+x ~/GitHub/r-config/R-devel/scripts/update-R-devel.sh
```

12. Run R-devel:

    From the commandline, simply do:

```
R-devel
```

Manually switching R versions for use with RStudio

RStudio will check for an environment variable, `RSTUDIO_WHICH_R`, which can be set to override using the installed system version of R.

RStudio desktop

```
export RSTUDIO_WHICH_R=/usr/local/bin/R-4.2.3
rstudio &
```

RStudio server

As `sudo` user:

```
umask 0022
echo "export RSTUDIO_WHICH_R=/usr/local/bin/R-3.6.3" > /etc/profile.d/rstudio.sh
```

References

- https://stackoverflow.com/a/24019938/1380598
- https://stat.ethz.ch/pipermail/r-sig-debian/2012-August/001937.html
- http://singmann.org/installing-r-devel-on-linux/
- https://github.com/rocker-org/rocker-versioned2
- https://hub.docker.com/r/rocker/drd/~/dockerfile/
- https://support.posit.co/hc/en-us/articles/218004217-Building-R-from-source
- https://community.rstudio.com/t/use-a-different-r-version-temporarily-in-rstudio/20848/8
- https://solutions.posit.co/envs-pkgs/environment-variables/
- https://github.com/r-lib/rig