Using multiple R versions on Linux

Alex M. Chubaty

December 6, 2016

Contents

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 reproducbility of old code/scripts, or to rerun old anasyses.

Using Docker

This is the preferred solution, as it is self-contained. Everytime 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 `packrat` 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 offical 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:3.2.2
   ```

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

   ```
   docker run -it --rm r-base:3.2.2
   ```

   * 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.

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

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.

Metal installation

As noted above, using Docker is likely a better solution, as this approach doesn't maintain separation between packages installed for different versions (except R-devel).

Older R versions

1. Install prerequisites for building R from source:

   ```
   sudo apt-get 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:

   ```
   cd ~/R/src/
   wget https://cran.r-project.org/src/base/R-3/R-3.2.2.tar.gz
   tar -xvzf R-3.2.2.tar.gz
   ```

4. Install:

   ```
   cd ~/R/src/R-3.2.2
   ./configure --prefix=/usr/local/lib/R-3.2.2 --enable-R-shlib --with-blas --with-lapack
   make
   sudo make install
   ```

5. Create symlink:

   ```
   sudo ln -s /usr/local/lib/R-3.2.2/bin/R /usr/local/bin/R-3.2.2
   ```

6. Run a specific R version:

   From the commandline, simply do:

   ```
   R-3.2.2
   ```

R-devel

1. Install prerequisites for building R from source:

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

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

   ```
   mkdir ~/R/src
   mkdir ~/R/src/R-devel-build
   ```

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. Create the installation script in `~/R/bin/build-R-devel.sh`:

```
## ~/R/bin/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 -j4"                               \
../r-devel/R/configure                        \
    --prefix=/usr/local/lib/R-devel           \
    --enable-R-shlib                          \
    --with-blas                               \
    --with-lapack                             \
```

```
    --with-readline                                 \
    --with-recommended-packages                     \
    --program-suffix=devel

make

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

6. Give the script execute permissions:

```
chmod a+x ~/R/bin/build-R-devel.sh
```

7. Make and install R-devel:

```
bash ~/R/bin/build-R-devel.sh

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

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

```
## ~/R/bin/R-devel.sh
#!/bin/bash
export R_LIBS_SITE=${R_LIBS_SITE-'/usr/local/lib/R-devel/lib/R/library:/usr/local/lib/R/site-librar
export PATH="/usr/local/lib/R-devel/bin:$PATH"
R "$@"
```

   Be sure to give the script execute permissions:

```
chmod a+x ~/R/bin/R-devel.sh
```

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

```
## ~/R/bin/Rscript-devel.sh
#!/bin/bash
export R_LIBS_SITE=${R_LIBS_SITE-'/usr/local/lib/R-devel/lib/R/library:/usr/local/lib/R/site-librar
export PATH="/usr/local/lib/R-devel/bin:$PATH"
Rscript "$@"
```

   Be sure to give the script execute permissions:

```
chmod a+x ~/R/bin/Rscript-devel.sh
```

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

```
sudo ln -s ~/R/bin/R-devel.sh /usr/local/bin/R-devel
sudo ln -s ~/R/bin/Rscript-devel.sh /usr/local/bin/Rscript-devel
```

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

    This could be turned into a `cron` job.

```
## ~/R/bin/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

# rebuild
```

```
cd  ~/R/bin
bash ./build-R-devel.sh

# make and install
cd ~/R/src/R-devel-build
sudo make install

# update R packages
echo "*** Installation complete. Updating R packages..."
Rscript-devel -e 'update.packages(ask = FALSE, lib.loc = .Library.site[1])'
echo "*** Done."
```

Be sure to give the script execute permissions:

```
chmod a+x ~/R/bin/update-R-devel.sh
```

12. Run R-devel:

    From the commandline, simply do:

    ```
    R-devel
    ```

References

- http://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://hub.docker.com/r/rocker/drd/~/dockerfile/
- https://support.rstudio.com/hc/en-us/articles/218004217-Building-R-from-source