Advanced R Package Installation and Loading

Brett J. Gall

06 August 2018

Overview

Currently, reproducible processes for installing and loading current and older versions of packages in R involve many dependencies or considerable manual coding and many approaches are not very flexible. For example, while `devtools::install_url()` and `devtool::install_version()` can try to install a specific version of a package or install from a URL, `devtools` (a) installs regardless of whether the package is already installed, (b) doesn't load the package, and (c) can't install/load multiple packages at once. This document aims to provide a flexible function others can use to install and load packages for reproducible research without relying on dependencies other than base R.

Motivation

To motivate this package, let's begin with the simplest scenario: we have a vector of packages and want to check if they are installed, install the packages if they are not already installed, then load the packages. This assumes we are satisfied with any version of the package. If a package is installed but is not the current version of the package, the package will be loaded without updating the version. We can accomplish this using the function below.

```r
# Define function for loading/installing a function
loadpkg <- function(toLoad){
  for(lib in toLoad){
    if(! lib %in% installed.packages()[ ,1]) {
      install.packages(lib, repos = 'http://cran.rstudio.com/')
      }
    suppressMessages(library(lib, character.only=TRUE))
  }
  }

# Here's are some working examples where we install some packages:
loadpkg(c("dplyr","margins","ggplot2"))
loadpkg(c("interplot","tidyr"))

# We don't even need the concatenate function c() if we have one package:
loadpkg("dplyr")
```

But what if we want to install a different version of a package? This is more involved as the install.packages() function in base R doesn't have anything like a `version` argument. Second, this will typically involve directing R to a URL where the source code is available. However, Windows machines typically won't be able to compile code from source without having some software (e.g. Rtools.exe) already installed. Third, some versions of packages may not be compatible with some versions of the R software and you simply won't be able to use them with your current version of R. More about this can be found via https://support.rstudio.com/hc/en-us/articles/219949047-Installing-older-versions-of-packages and https://stackoverflow.com/questions/22673474/how-to-install-doredis-package-version-1-0-5-into-r-3-0-1-on-windows/22673831#22673831

Ideally, we write a very flexible function allowing users to provide (i) the name of a package as found in our above loadpkg() function, (ii) the name of the package *and* the version of the package if we want an older version of the package, and/or (iii) a URL to the package on CRAN-like repositories. The function then installs a (specific version of a package) package if it is not already installed and loads all packages once they are installed.

How to design the function?

Suppose we want to load (and install if not installed) version 3.4.0 of ggplot2 and version 2.0.1.2 of dplyr. Let's first assume version 3.4.0 of ggplot2 is the current version of the package. We'll want something that looks for whether the package is installed and installs the package using the standard install.packages() function (i.e. our loadpkg() function above) if the desired package version is current or altenratively a different function if the desired package version is not current. If the package is installed and we are indifferent about the version of the package, we'll load the package and won't update it. In psuedo-code, we want something like: load_pkgs(pkgs = c("ggplot2","dplyr"), versions = c(NA,"2.0.1.2"). This would then lead to loadpkg("ggplot2) because is.na(versions) == TRUE for the first element and something like load_version("dplyr","2.0.1.2") because is.na(versions) == FALSE for the second element.

This suggests we might create an if() statement to check whether each element of the versions vector is.na() == TRUE and use our loadpkg() function for those packages and other functions for is.na() == FALSE packages. However, we one issue with doing so is that we want users to be able to provide a URL in place of the package name *and* leave the version as NA. So, we'll need to add one other option: if the package name is a URL, use some other function to figure out the package name, the version name, then install/load accordingly.

This raises the question: if we have a URL for downloading a package, how do we know the package's version and name? If we didn't have it, we would just repeatedly install the same package even if it was previously installed. That is inefficient. Conveniently, we can use CRAN naming conventions of files to obtain this information. Head over to the CRAN package archive and scroll down to the ggplot2 directory. Click on it (alternatively, here's the URL). We now see many files containing different versions of the ggplot2 package. A sample file is here. If one goes to a different directory one will similarly find many .tar.gz files.

The key feature of this file's name comes after the last slash "/". For all packages conforming with CRAN standards (hypothetically this should mean all R packages), the package name comes after the last slash then is followed by an underscore, the version number, and the file extension .tar.gz. We can therefore take any character input, check if it ends with .tar.gz or contains the string http://, then find the package name by looking at all characters after the last slash and before the last underscore. All characters between the underscore and the .tar.gz ending will contain the version number. For example, let's look at the archive for ggplot2. Each file begins with the name of the package (ggplot2), an underscore and package version follows, then the file names end with .tar.gz.

Getting started

Let's start creating our function. We want the function to take two argument: a vector of packages names or URLs and a vector of version names. We'll assign a default vector of NAs to the version argument in case we don't want to specify the package version. Defining the function's arguments should look like this:

```r
# Function name and arguments
loadpkg <- function(package, version = rep(NA, length(package))){ }
```

We now want to add the three different scenarios we laid out above.

Input: package name but no version

First, let's write out the statement to check if the package name is provided rather than a URL *and* no version is specified:

```r
# Create a sample input for our functions below to test them
package <- c("ggplot2")
version <- NA

# Check if package contains a URL:
url_test <- ifelse(length(grep("http://", package)) +
                   length(grep("https://", package)) > 0,
                   TRUE,
                   FALSE)

# If package name provided but no version, install
# package if not installed. Load it.
if(url_test == FALSE & is.na(version) == TRUE){
     if(!package %in% installed.packages()[ ,1]) {
      install.packages(package, repos = 'http://cran.rstudio.com/')
       library(package)
     }
  }
```

Input: package name and version

We next want to write the code for the condition where the user provides a package name (rather than a URL) and also a version number. If the version number matches a currently installed version, there's no need to do anything! Otherwise, install the specified version.

```r
# If a package name is provided and a version number is
# provided, check if the version is the most recent on CRAN.
# If so, install as usual if not already installed.
# Otherwise,install regardless of whether it is installed
# (as a different version).
if(url_test == FALSE & is.na(version) == FALSE){

  # Create a URL for downloading/installing the package from CRAN
  package_url <- paste0("http://cran.r-project.org/src/contrib/Archive/",
                    package,"/",package,"_",version,".tar.gz")

  # If the package isn't installed, install the specified version:
  if(!package %in% installed.packages()[ ,1]){
    install.packages(package_url, repos = NULL, type = "source")
  }

  # Check if the package is installed. If it is installed, check
  # if the installed version is the same as the specified version.
  # If so, do nothing. Otherwise, try to install the specified
  # version from CRAN.
  if(package %in% installed.packages()[ ,1]){
    if(version != installed.packages()[package,3]){
          install.packages(package_url, repos = NULL, type = "source")
    }
  }

  # Load the package
  library(package)
}
```

Input: URL, no version

Now we'll address the final scenario where the user provides a URL only. We want to extract the package name and the version number then check if it's installed and install if it isn't.

```r
# If URL is provided, check if the specified
# package and version is already installed. If so,
# do nothing. Otherwise, install it.
if(url_test == TRUE & is.na(version) == TRUE){

  # Extract the package name and version from the URL
  pkg_name <- sub(".tar.*", "", strsplit(basename(package), "_")[[1]])[1]
  pkg_version <- sub(".tar.*", "", strsplit(basename(package), "_")[[1]])[2]

  # If the package isn't installed, install the specified version:
  if(!pkg_name %in% installed.packages()[ ,1]){
    install.packages(package_url, repos = NULL, type = "source")

    # Load package
    library(pkg_name)
  }

  # If the package is installed, check if the version matches
  # the currently loaded version. Install if not installed.
  if(package %in% installed.packages()[ ,1]){
    if(pkg_version != installed.packages()[package,3]){
          install.packages(package_url, repos = NULL, type = "source")
    }

    # Load package
    library(pkg_name)
  }
}
```

Input: URL and version

Finally we can add an error message if the user provides a URL and also a version number, since we only want users to provide a version number if they are providing the name of the package.

```r
if(url_test == TRUE & is.na(version) == FALSE){
  stop("If you provide a URL, you cannot provide a
       version or must provide a NA value to the
       version argument. The default value is NA.")
  }
```

Putting it all together

Now that we have the individual chunks of code for each potential input type, we can put it all together into a single function.

```r
# Function is named pkg_function. Takes one function
# as an input
pkg_function <- function(package, version = NA){

  # Check if package contains a URL:
  url_test <- ifelse(length(grep("http://", package)) +
                     length(grep("https://", package)) > 0,
                     TRUE,
                     FALSE)

  # If package name provided but no version, install
  # package if not installed.
  if(url_test == FALSE & is.na(version) == TRUE){
    if(!package %in% installed.packages()[ ,1]){
      install.packages(package, repos = 'http://cran.rstudio.com/')
    }

    # Load package
    library(package)
  }

  # If a package name is provided and a version number is
  # provided, check if the version is the most recent on CRAN.
  # If so, install as usual if not already installed.
  # Otherwise,install regardless of whether it is installed
  # (as a different version).
  if(url_test == FALSE & is.na(version) == FALSE){

    # Create a URL for downloading/installing the package from CRAN
    package_url <- paste0("http://cran.r-project.org/src/contrib/Archive/",
                          package,"/",package,"_",version,".tar.gz")

    # If the package isn't installed, install the specified version:
    if(!package %in% installed.packages()[ ,1]){
      install.packages(package_url, repos = NULL, type = "source")
      }

    # Check if the package is installed. If it is installed, check
    # if the installed version is the same as the specified version.
    # If so, do nothing. Otherwise, try to install the specified
    # version from CRAN.
    if(package %in% installed.packages()[ ,1]){
      if(version != installed.packages()[package,3]){
        install.packages(package_url, repos = NULL, type = "source")
      }
    }
    # Load package
    library(package)
  }
  # If URL is provided, check if the specified
  # package and version is already installed. If so,
  # do nothing. Otherwise, install it.
  if(url_test == TRUE & is.na(version) == TRUE){

    # Extract the package name and version from the URL
    pkg_name <- sub(".tar.*", "", strsplit(basename(package), "_")[[1]])[1]
    pkg_version <- sub(".tar.*", "", strsplit(basename(package), "_")[[1]])[2]

    # If the package isn't installed, install the specified version:
    if(!pkg_name %in% installed.packages()[ ,1]){
      install.packages(package_url, repos = NULL, type = "source")
      library(pkg_name)
      }

    # If the package is installed, check if the version matches
    # the currently loaded version. Install if not installed.
    if(package %in% installed.packages()[ ,1]){
      if(pkg_version != installed.packages()[package,3]){
        install.packages(package_url, repos = NULL, type = "source")
        library(pkg_name)
      }
    }
  }
}
```

Appyling the function to multiple packages

We now want to apply this function - which accepts a single package name and version as an input - to each element of a vector of package names and a vector of versions. Although one could vectorize the below, we'll loop through each element of the input vectors using a for loop.

```
# Wrap a for-loop around pkg_function to apply the function to
# each element of a vector of package names and versions.
loadpkg <- function(packages, versions = rep(NA,length(packages))){
  for(i in 1:length(packages)){
    pkg_function(package = packages[i], version = version[i])
    }
  }
```

The final product: loadpkg()

That's it! In case it is useful, here is our function in a single code chunk.

```
# Wrap a for-loop around pkg_function to apply the function to
# each element of a vector of package names and versions.
loadpkg <- function(packages, versions = rep(NA,length(packages))){

  # Function is named pkg_function. Takes one function
  # as an input
  pkg_function <- function(package, version = NA){
```

```r
# Check if package contains a URL:
url_test <- ifelse(length(grep("http://", package)) +
                   length(grep("https://", package)) > 0,
                   TRUE,
                   FALSE)

# If package name provided but no version, install
# package if not installed. Load it.
if(url_test == FALSE & is.na(version) == TRUE){
  if(!package %in% installed.packages()[ ,1]){
    install.packages(package, repos = 'http://cran.rstudio.com/')
    library(package)
  }
}

# If a package name is provided and a version number is
# provided, check if the version is the most recent on CRAN.
# If so, install as usual if not already installed.
# Otherwise,install regardless of whether it is installed
# (as a different version).
if(url_test == FALSE & is.na(version) == FALSE){

  # Create a URL for downloading/installing the package from CRAN
  package_url <- paste0("http://cran.r-project.org/src/contrib/Archive/",
                        package,"/",package,"_",version,".tar.gz")

  # If the package isn't installed, install the specified version
  # and load it.
  if(!package %in% installed.packages()[ ,1]){
    install.packages(package_url, repos = NULL, type = "source")
    library(package)
    }

  # Check if the package is installed. If it is installed, check
  # if the installed version is the same as the specified version.
  # If so, do nothing. Otherwise, try to install the specified
  # version from CRAN. Load it.
  if(package %in% installed.packages()[ ,1]){
    if(version != installed.packages()[package,3]){
      install.packages(package_url, repos = NULL, type = "source")
      library(package)
    }
  }
}
# If URL is provided, check if the specified
# package and version is already installed. If so,
# do nothing. Otherwise, install it.
if(url_test == TRUE & is.na(version) == TRUE){

  # Give a warning Rtools must be installed for windows
  warning("You have provided a URL to the package source code.
          If you are using Windows, make sure Rtools is installed
          to compile the code!")
      # Extract the package name and version from the URL
      pkg_name <- sub(".tar.*", "", strsplit(basename(package), "_")[[1]])[1]
      pkg_version <- sub(".tar.*", "", strsplit(basename(package), "_")[[1]])[2]

      # If the package isn't installed, install the specified version
      # and load it.
      if(!pkg_name %in% installed.packages()[ ,1]){
        install.packages(package_url, repos = NULL, type = "source")
        library(pkg_name)
        }

      # If the package is installed, check if the version matches
      # the currently loaded version. Install if not installed.
      if(package %in% installed.packages()[ ,1]){
        if(pkg_version != installed.packages()[package,3]){
          install.packages(package_url, repos = NULL, type = "source")
          library(pkg_name)
        }
      }
    }
  }

  # Iterate through the vector of package names/urls
  for(i in 1:length(packages)){
    pkg_function(package = packages[i], version = version[i])
  }
}
```

Examples

```
# Install a single package, no version specified
loadpkg(package = "interplot")

# Install multiple packages, no version specified
loadpkg(package = c("interplot", "ggplot2"))

# Install a single package, old version specified
loadpkg(package = "interplot", version = "0.1.0.1")

# Install multiple packages, old versions specified
loadpkg(package = c("interplot", "ggplot2"), version = c("0.1.0.1", "2.1.1"))

# Install from a URL
loadpkg(package = "https://cran.r-project.org/src/contrib/Archive/ggplot2/ggplot2_0.9.3.tar.gz")
```

```
## Warning in pkg_function(package = packages[i], version = version[i]): You have provided a URL to the
##                 If you are using Windows, make sure Rtools is installed
##                 to compile the code!
```

```
# Install from a URL, with version specified (should throw an error)
loadpkg(package = "https://cran.r-project.org/src/contrib/Archive/ggplot2/ggplot2_0.9.3.tar.gz",
        version = "2.0")
```

```
## Warning in pkg_function(package = packages[i], version = version[i]): You have provided a URL to the
##               If you are using Windows, make sure Rtools is installed
##               to compile the code!
```