

R Packages for a Reproducible Workflow

Daniela Palleschi

2023-08-30

Table of contents

1	here package	1
2	pacman package	2
3	renv package	2
3.1	Activate or init	3
3.2	Snapshot	3
3.3	Restore	3
3.4	Upgrade	3
3.5	Update and Hydrate	4
4	usethis package	4
5	gitcreds package	5

1 here package

- uses file path relative to the current project

```
# install here package
install.packages("here")

# read in a csv file
readr::read_csv(here::here("folder", "subfolder", "subsubfolder", "file.csv"))
# or
readr::read_csv(here::here("folder/subfolder/subsubfolder/file.csv"))
```

2 pacman package

- `p_load()` function checks if listed packages are already installed
 - if yes, loads them (as with `library()`)
 - if no, installs and then loads them (as long as they're CRAN packages!)

```
# install pacman
install.packages("pacman")

# install and/or load packages
pacman::p_load(here,
               renv,
               kableExtra)
```

3 renv package

The `renv` package stores your project-local environment. That is, it creates a time capsule of the R and package versions that you use *within an R Project*.

The benefits:

- your output is not dependent on whichever project version you currently have stored on your machine
 - e.g., if you come back to an analysis after a long time, the output should still be the same even if a certain package version has deprecated a function your analysis uses.
- makes the project *reproducible* for collaborators as well, who will likely have some different package versions (or will have missing packages) on their machine

The hard-to-get-your-head-around (imo):

- with each new R Project with a `renv.lock` file, you need to install all the necessary packages again (even if they're on your machine)
 - this is because each `renv` doesn't look beyond your project folder!
- remembering to update the `renv.lock` file frequently!

Running `renv` functions

Remember to always run `renv` functions *in the console*! You do not want e.g., `renv` initialising a new `renv.lock` file every time you render your documents.

3.1 Activate or init

- to initialise a new *project-local environment*:

```
renv::init()
```

3.2 Snapshot

- update `renv.lock` file

```
renv::snapshot()
```

After updating the `renv.lock` file, remember to commit/push these changes to git (if you're using it)!

Version control

After updating the `renv.lock` file (i.e., running `renv::snapshot()`), remember to commit/push these changes to git (if you're using git)! Otherwise, your `renv.lock` file will be outdated compared to your output.

3.3 Restore

- will restore your project to the most recent `renv.lock` file versions
 - this step follows `snapshot()`, which updates the `renv.lock` file
- very useful after updating R!

```
renv::restore()
```

3.4 Upgrade

- to update the `renv` package:

Table 1: Summary of useful commands from the usethis package

command	function
<code>git_token_help()</code>	print credential situation
<code>git_sitrep()</code>	print more verbose credential information
<code>git_vaccinate()</code>	add appropriate files to .gitignore (remember to stage, commit, and push)
<code>browse_github()</code>	view project GitHub
<code>browse_project()</code> / <code>__package()</code>	visit project-related pages
<code>browse_github_actions()</code>	view overview of GitHub actions
<code>git_credentials()</code>	print credentials
<code>create_github_token()</code>	create a Personal Access Token (PAT)
<code>gitcreds_set()</code>	add your PAT

```
# upgrade renv version
renv::upgrade()
```

3.5 Update and Hydrate

```
# updates all packages (stored in renv.lock) in the renv cache
renv::hydrate(update = "all")

# update should have no effect now, but doesn't hurt to check
renv::update()

# now take a snapshot with the updated packages
renv::snapshot()
```

Version control (repeated)

After updating the `renv.lock` file (i.e., running `renv::snapshot()`), remember to commit/push these changes to git (if you're using git)! Otherwise, your `renv.lock` file will be outdated compared to your output.

4 usethis package

The `usethis` package automates project set-up and development, and is also useful when creating a package (see the package [Documentation](#) for more info).

Table 2: Summary of useful commands from the gitcreds package

command	function
gitcreds_get()	print git/GitHub credentials
gitcreds_set()	set/replace credentials
gitcreds_delete()	delete credentials

5 gitcreds package

Has some similar functionality to **usethis** with regards to using git/GitHub in an R project. See the package [GitHub page](#) for more information.