renv

Reproducible Environments for R

A Practical Guide to Professional R Development

Reproducible
Projects

Production
Ready

Team
Collaboration

Dr. Saad Laouadi

AI Expert & Data Science Educator

Independent Consultant

Version 1.0 | December 17, 2025

Professional R Development Series

Contents

1 What is renv? 2
1.1 The Problem . . . 2
1.2 The Solution . . . 2
1.3 What renv Does . . . 2

2 Getting Started 3
2.1 Installation . . . 3
2.2 Initialize Your First Project . . . 3
2.3 What Gets Created . . . 3
2.4 Project Structure . . . 3

3 Basic Workflow 4
3.1 The renv Cycle . . . 4
3.2 The 4-Step Process . . . 4

4 Essential Commands 6
4.1 Understanding renv::status() . . . 6

5 Common Workflows 7

6 Best Practices 8
6.1 Complete .gitignore Template . . . 8

7 Troubleshooting 10

8 Quick Reference 11

1 What is renv?

renv locks your R package versions so your code works everywhere, every time.

1.1 The Problem

Your R code runs perfectly today. Six months later or on a colleague's machine it breaks.
Root cause: Package versions changed. No record of what worked.

1.2 The Solution

Without renv

- Shared global library
- Updates break old code
- No version control
- "Works on my machine"

With renv

- Isolated per project
- Versions locked
- Fully reproducible
- Works everywhere

1.3 What renv Does

- **Captures** exact package versions
- **Records** them in `renv.lock`
- **Restores** identical environment anywhere
- **Isolates** projects from each other

2 Getting Started

2.1 Installation

```
install.packages("renv")
```

2.2 Initialize Your First Project

Step 1: Open your R project

Step 2: Run initialization

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

2.3 What Gets Created

File/Folder	Purpose
`renv/library/`	Isolated package storage
`renv.lock`	Version lockfile (commit this!)
`.Rprofile`	Auto-activates renv
`renv/activate.R`	Activation script

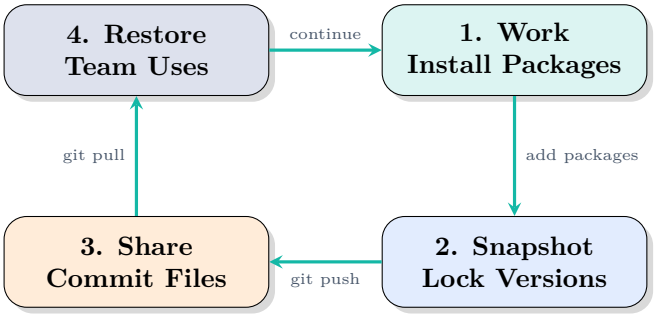
2.4 Project Structure

```
my_project/
 renv/
  library/ # Packages (don't commit)
  activate.R # Startup script
 renv.lock # Version record
 .Rprofile # Auto-loads renv
 analysis.R
```

After `renv::init()`, install packages normally with `install.packages()`. They'll go to `renv/library/` automatically.

3 Basic Workflow

3.1 The renv Cycle



3.2 The 4-Step Process

Step 1: Install Packages

```
install.packages(c("dplyr", "ggplot2", "tidyr"))
```

Packages install to `renv/library/` (isolated from global library).

Step 2: Snapshot Dependencies

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

Locks package versions in `renv.lock`.

Step 3: Share Project
Commit to git:

- `renv.lock`
- `.Rprofile`
- `renv/activate.R`

Add to .gitignore:

```
renv/library/
```

Step 4: Restore on Team Machine

```
# Collaborator clones repo, then:
renv::restore()
```

Installs exact versions from `renv.lock`. Environment matches perfectly.

Run `renv::status()` before `snapshot()` to verify what will be locked.

4 Essential Commands

```
# Initialize renv in current project
renv::init()


# Check status (are packages in sync?)
renv::status()


# Snapshot current package versions
renv::snapshot()


# Restore packages from lockfile
renv::restore()


# Update packages
renv::update()


# Install a package
install.packages("package_name")


# Remove unused packages
renv::clean()
```

4.1 Understanding renv::status()

Run `renv::status()` to check if your project is in sync.
It tells you:

- Packages installed but not in lockfile
- Packages in lockfile but not installed
- Packages with different versions

```
# Example output
> renv::status()


The following package(s) are installed but not recorded in the lockfile:
tidyr [1.3.0]


Use renv::snapshot() to save the state of your library.
```

5 Common Workflows

Scenario 1: New Project

```
renv::init()
install.packages(c("dplyr", "ggplot2"))
renv::snapshot()
```

Done. Start coding.

Scenario 2: Join Existing Project

```
# After cloning repository:
renv::restore()
```

renv activates automatically. Packages install from `renv.lock`.

Scenario 3: Add Package

```
install.packages("readr")
renv::status() # Verify
renv::snapshot() # Lock it
```

Commit updated `renv.lock`.

Scenario 4: Update Packages

```
renv::update("dplyr") # Update one
# OR
renv::update() # Update all

renv::snapshot() # Lock new versions
```

Updates can break code. Test before snapshotting.

6 Best Practices

Golden Rule

`renv::snapshot()` after every package change

1. Start Every Project with renv

```
renv::init() # First thing, not last
```

Don't wait until deployment.

2. Version Control Strategy

Commit

- `renv.lock`
- `.Rprofile`
- `renv/activate.R`

Ignore

- `renv/library/`
- `renv/staging/`
- `renv/local/`

3. Check Status Before Committing

```
renv::status() # Always verify sync
```

Catches issues before they reach your team.

4. Document renv in README

```
# README.md
This project uses renv.
To restore:  renv::restore()
```

Helps new collaborators get started.

5. Test Restore Periodically

```
# On fresh machine or Docker container
git clone repo
renv::restore()
```

Verify your lockfile works before others depend on it.

6.1 Complete .gitignore Template

```
# renv - ignore these
renv/library/
renv/local/
renv/cellar/
renv/staging/
```

```
renv/python/
```

This `.gitignore` keeps repos clean. Commit small files, ignore large libraries.

7 Troubleshooting

Issue 1: Installation Fails During Restore

```
Error:  installation of package 'X' had non-zero exit status
```

Fix:

- Install system dependencies (Linux: `apt-get`, Mac: `brew`)
- Update renv: `install.packages("renv")`
- Install manually: `install.packages("package_name")`

Issue 2: Packages Out of Sync

```
> renv::status()
The following packages are out of sync...
```

Fix:

Update lockfile:

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

Revert to lockfile:

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

Issue 3: renv Not Activating

Fix: Verify `.Rprofile` contains:

```
source("renv/activate.R")
```

If missing, re-initialize: `renv::init()`

Issue 4: Team Can't Restore

Common causes:

- Missing `renv.lock` (forgot to commit)
- Outdated renv version
- Firewall blocking downloads

Fix: Update renv, try again

```
install.packages("renv")
renv::restore()
```

8 Quick Reference

renv Command Reference

Command	Purpose
`renv::init()`	Initialize new project
`renv::status()`	Check sync status
`renv::snapshot()`	Lock current versions
`renv::restore()`	Install from lockfile
`renv::update()`	Update packages
`renv::clean()`	Remove unused packages
`install.packages()`	Add new package

Daily Workflow

1. Work & install packages
2. Run `renv::status()` to verify
3. Run `renv::snapshot()` to lock
4. Commit `renv.lock` to git

Commit

- `renv.lock`
- `.Rprofile`
- `renv/activate.R`

Ignore

- `renv/library/`
- `renv/staging/`
- `renv/local/`

Core Principle

`renv.lock` = Single source of truth

Lock versions. Share lockfile. Restore anywhere.