Zehui Yin

PhD Student in Geography at SEES

- **Email:** yinz39@mcmaster.ca
  - Please use your McMaster email address.
  - Please put the course code *ENVSOCTY 4GA3* in the subject line.
  - Please include your name and student number in the body of the email.
  - I will try to reply within 24 hours (please expect longer delays during weekends/holidays).
- **Personal Website:** zehuiyin.github.io

- **Research Interests:** Spatial Analysis, Econometrics, Machine Learning, Transportation Planning, Public Transit, and Micromobility

Agenda for today

- Introduction to basic concepts for coding in R
- Getting a flavour of R syntax and style
- Setting up R and reproducible environment on your personal computer
  - Instructions for both Windows and Mac systems
  - A demo on a Windows machine

    If you are using Linux, I expect you already have the technical skills to install R on your own computer :)

Lab slides

- You can access all the lab slides by scanning the QR code or by visiting the URL below directly.
  - zehuiyin.github.io/ENVSOCTY4GA3
- Slides from previous years are also available. You can take a peek, though I'll make some updates this year—likely not major ones.

R and RStudio

- R is a free and open-source programming language for statistical computing and graphics.
- RStudio is an integrated development environment (IDE) for coding in R.
  - An IDE is a set of tools that helps you code.
- We use RStudio to write our R codes.

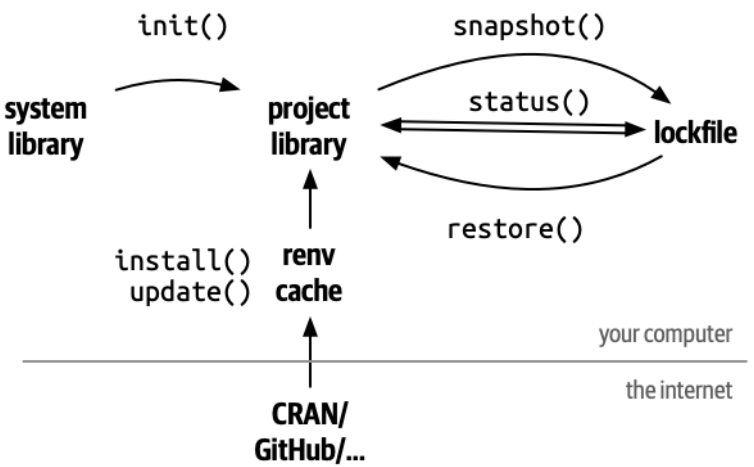
R packages

- R packages are the fundamental units of reproducible R code.
- They can include functions, data, or both, along with documentation.
- Think of them as plug-ins that enhance the functionality of existing software.
- For example, web browser extensions like ad blockers add additional features that the original browser doesn’t have.
- In this course, we use additional R packages to work with geospatial data in R.

Reproducible environment

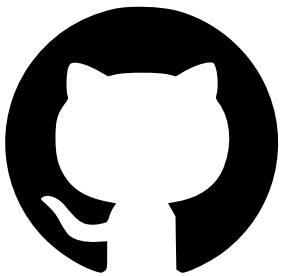
- An environment is the system where a program is run, including hardware and software such as operating system dependencies, programming language, packages, their configuration, and versions.
- Just as running 1000 meters affects individuals differently, running code on different computers or with different package versions can produce varied results.
- A reproducible environment ensures that everyone gets the same result by keeping the environment consistent.
  - Ideally, we would run the same program on the same system with the same software versions for maximum reproducibility.
  - In practice, we typically focus on ensuring that key software—such as R and R package versions—is the same.

renv package



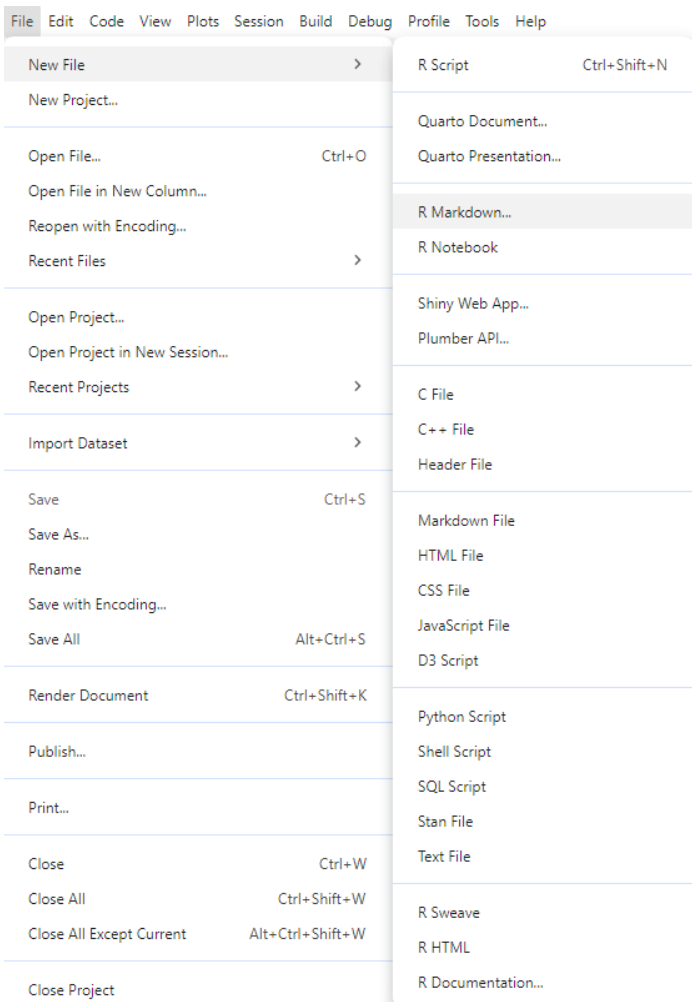
- renv is an R package that helps create reproducible environments for R projects.
- It records the R version and all R packages along with their versions in a lockfile.
  - A lockfile is a text file that stores all the environment information.

Code hosting and Github



- Code hosting involves storing code online to facilitate sharing, management, and collaboration with others.
- One of the most popular code hosting platforms is GitHub (owned by Microsoft).
- Both the textbook and the companion R package used in this course are hosted on GitHub.
- GitHub is like a cloud drive (similar to OneDrive or Dropbox) but specialized for storing code (more specifically plain text files), including R scripts.

R Markdown vs. R



- R Markdown is a file format that combines R code, its results (after knitting), and accompanying text.
- It uses the file extension `.Rmd` and essentially is a plain text file integrating markdown and R.
  - When you knit the file, the code is executed, the results are inserted, and the document is rendered into formats such as PDF, HTML, or other report file types.
- You can start by creating an `Rmd` file on the lab computer.

R Markdown syntax

```
---
title: "Untitled"
author: "Zehui Yin"
output: html_document
---

# Heading level 1
## Heading level 2
text...text...text
**bold**   __bold__
*italic*   _italic_

```{r}
print("Hello world!)
```
```

① **YAML header**: stores settings or meta information

② **Markdown text**: contains plain text in markdown format.

③ **R code chunk**: contains R code to be executed

Markdown syntax

Basic Syntax

These are the elements outlined in John Gruber's original design document. All Markdown applications support these elements.

Element	Markdown Syntax

Heading	`# H1`
	`## H2`
	`### H3`

Bold	`**bold text**`
Italic	`*italicized text*`
Blockquote	`> blockquote`

Ordered List	`1. First item`
	`2. Second item`
	`3. Third item`

Unordered List	`- First item`
	`- Second item`
	`- Third item`

Code	`` `code` ``
Horizontal Rule	`---`
Link	`[title](https://www.example.com)`
Image	`![alt text](image.jpg)`

- Markdown is a simple and easy-to-use markup language for formatting documents.
  - Not for computational purposes.
  - Only for text formatting/ typesetting.
- For basic markdown sytanx refer to: www.markdownguide.org/basic-syntax
- Markdown cheatsheet: www.markdownguide.org/cheat-sheet

R basics: arithmetic operations

You can start by trying out R on the lab computer. Later, we'll set it up on your personal computer.

R can be used as a calculator, using intuitive symbols for these operations:

```
1 + 5
```

```
[1] 6
```

```
8 - 3
```

```
[1] 5
```

```
3 * 4
```

```
[1] 12
```

```
9 / 3
```

```
[1] 3
```

R basics: assigning values

One of the cornerstones of programming languages is assignment. You can assign a value/object to a name using `<-` (suggested R style) or `=` ("Python" style).

```
a <- 1
b <- 3
a + b
```

```
[1] 4
```

```
c = 7
d = 5
c * d
```

```
[1] 35
```

R basics: built-in functions

R comes with many built-in functions. The calling syntax is `function(parameter1, parameter2, ...)`. Additionally, with extra R packages, there are even more functions you can use.

```r
values <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
sum(values)
```

```
[1] 55
```

```r
mean(values)
```

```
[1] 5.5
```

```r
library(MASS)
# integrate the sin function from 0 to pi.
area(sin, 0, pi)
```

```
[1] 2
```

R basics: indexing

Indexing is the process of selecting specific values from an object based on their index location. Whenever you see `[]` or `$` in R, some form of indexing is happening.

```r
v <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

1 2 3 4 5 6 7 8 9 10

```r
v[2]
```

```
[1] 2
```

```r
v[2:4]
```

```
[1] 2 3 4
```

```r
v[c(TRUE, T, T, T, T, TRUE,
    FALSE, F, FALSE, F)]
```

```
[1] 1 2 3 4 5 6
```

R basics: indexing

```r
df <- data.frame(col1 = c(1, 2, 3),
                 col2 = c(4, 5, 6))
```

col1	col2
1	4
2	5
3	6

```r
df[1, 2]
```

```
[1] 4
```

```r
df[, "col2"]
```

```
[1] 4 5 6
```

```r
df$col1
```

```
[1] 1 2 3
```

R basics: flow control

Flow control is an important component of any programming language. In R, the `if-else` statement and loops work as follows:

```r
x <- 6
if (x > 5) {
  print("Greater than 5")
} else {
  print("Less or equal to 5")
}
```

```
[1] "Greater than 5"
```

```r
for (i in 1:3) {
  print(i)
}
```

```
[1] 1
[1] 2
[1] 3
```

R basics: custom functions

To define your own function in R, you can use the following syntax. Note that the last line of code is automatically returned by R, though a “Python” style return statement is also valid in R.

```r
add <- function(a, b) {
  a + b
}

add(1, 4)
```

```
[1] 5
```

```r
add <- function(a, b) {
  return(a + b)
}

add(1, 4)
```

```
[1] 5
```

Download R version 4.4.2

mirror.csclub.uwaterloo.ca/CRAN

CRAN
Mirrors
What's new?
Search
CRAN Team

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Task Views
Other

Documentation
Manuals
FAQs
Contributed

Donations
Donate

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux (Debian, Fedora/Redhat, Ubuntu)
- Download R for macOS
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

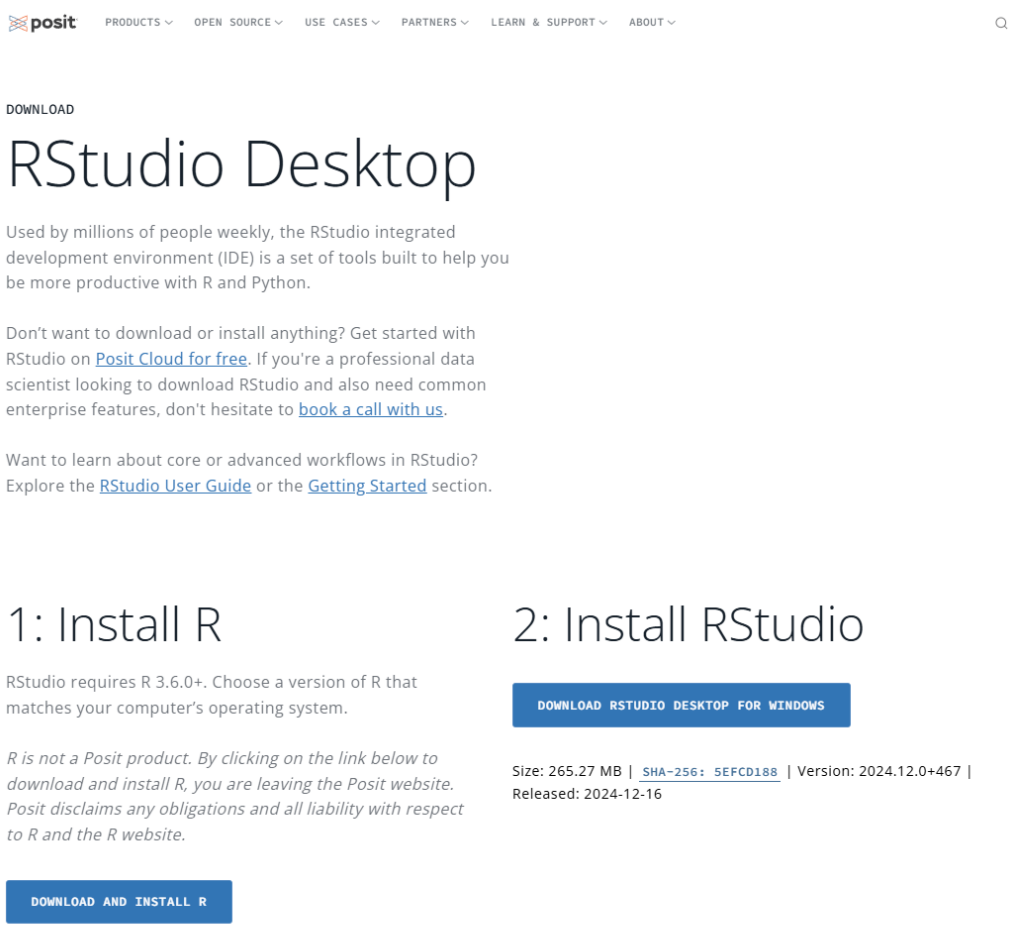
Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2024-10-31, Pile of Leaves) R-4.4.2.tar.gz, read what's new in the latest version.
- The CRAN directory src/base-prerelease contains R alpha, beta, and rc releases as daily snapshots in time periods before a planned release.
- Between releases, the same directory src/base-prerelease contains snapshots of current patched and development versions.
  Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Alternatively, daily snapshots are available here.
- Source code of older versions of R is available here.
- Contributed extension packages.

Download RStudio

posit.co/download/rstudio-desktop



Restoring the environment

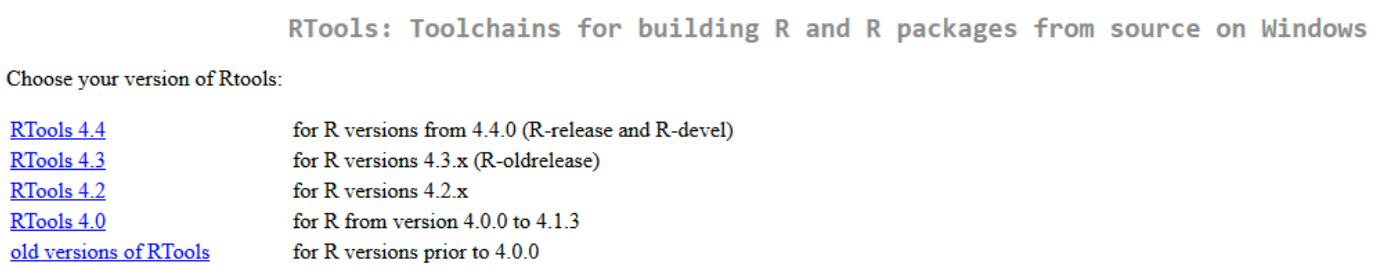
Download the `Applied-Spatial-Statistics` zip file from Avenue (or directly from GitHub: github.com/paezha/Applied-Spatial-Statistics) and **unzip** it. You should then have a folder with the following structure:

```
Applied-Spatial-Statistics/
├── renv/
├── .gitignore
├── .Rprofile
├── Applied-Spatial-Statistics.Rproj
├── README.md
├── README.Rmd
└── renv.lock
```

- Double-click the `Applied-Spatial-Statistics.Rproj` file to open the R project.

RTools 4.4 for Windows users

mirror.csclub.uwaterloo.ca/CRAN/bin/windows/Rtools



RTools: Toolchains for building R and R packages from source on Windows

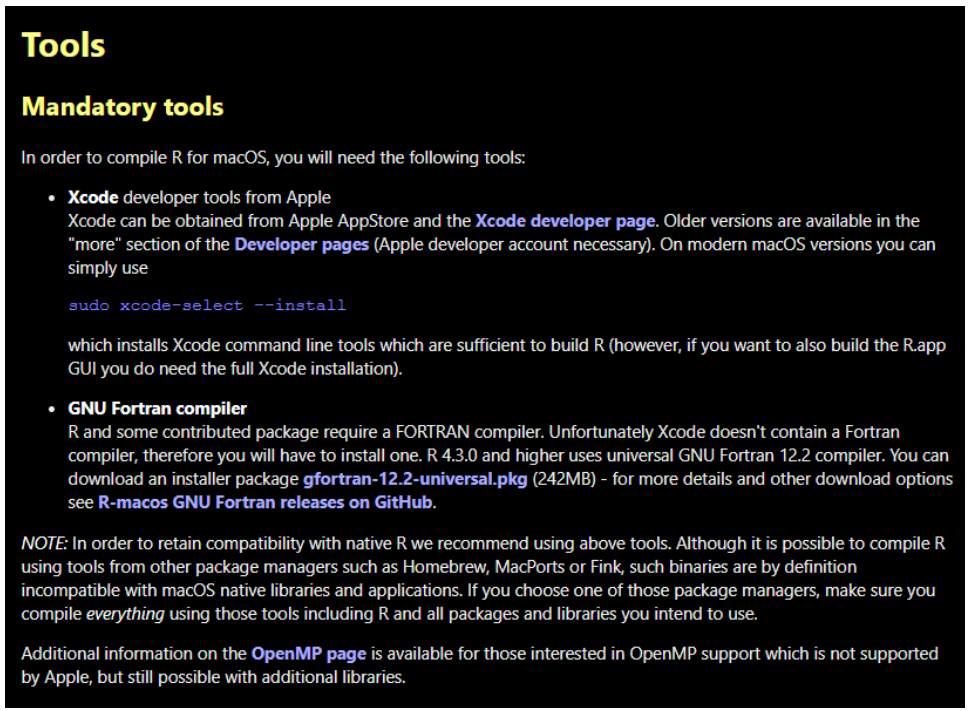
Choose your version of Rtools:

RTools 4.4	for R versions from 4.4.0 (R-release and R-devel)
RTools 4.3	for R versions 4.3.x (R-oldrelease)
RTools 4.2	for R versions 4.2.x
RTools 4.0	for R from version 4.0.0 to 4.1.3
old versions of RTools	for R versions prior to 4.0.0

- Ensure you download the `Rtools` version that matches your installed R version.
- Rtools is a set of programs required on **Windows** to build R packages from source.
- Note: If you are using Mac or Linux, Rtools is **not** required.

Xcode and GNU Fortran for Mac users

mac.r-project.org/tools



Tools

Mandatory tools

In order to compile R for macOS, you will need the following tools:

- **Xcode** developer tools from Apple
  Xcode can be obtained from Apple AppStore and the **Xcode developer page**. Older versions are available in the "more" section of the **Developer pages** (Apple developer account necessary). On modern macOS versions you can simply use

  ```
  sudo xcode-select --install
  ```

  which installs Xcode command line tools which are sufficient to build R (however, if you want to also build the R.app GUI you do need the full Xcode installation).

- **GNU Fortran compiler**
  R and some contributed package require a FORTRAN compiler. Unfortunately Xcode doesn't contain a Fortran compiler, therefore you will have to install one. R 4.3.0 and higher uses universal GNU Fortran 12.2 compiler. You can download an installer package **gfortran-12.2-universal.pkg** (242MB) - for more details and other download options see **R-macos GNU Fortran releases on GitHub**.

NOTE: In order to retain compatibility with native R we recommend using above tools. Although it is possible to compile R using tools from other package managers such as Homebrew, MacPorts or Fink, such binaries are by definition incompatible with macOS native libraries and applications. If you choose one of those package managers, make sure you compile *everything* using those tools including R and all packages and libraries you intend to use.

Additional information on the **OpenMP page** is available for those interested in OpenMP support which is not supported by Apple, but still possible with additional libraries.

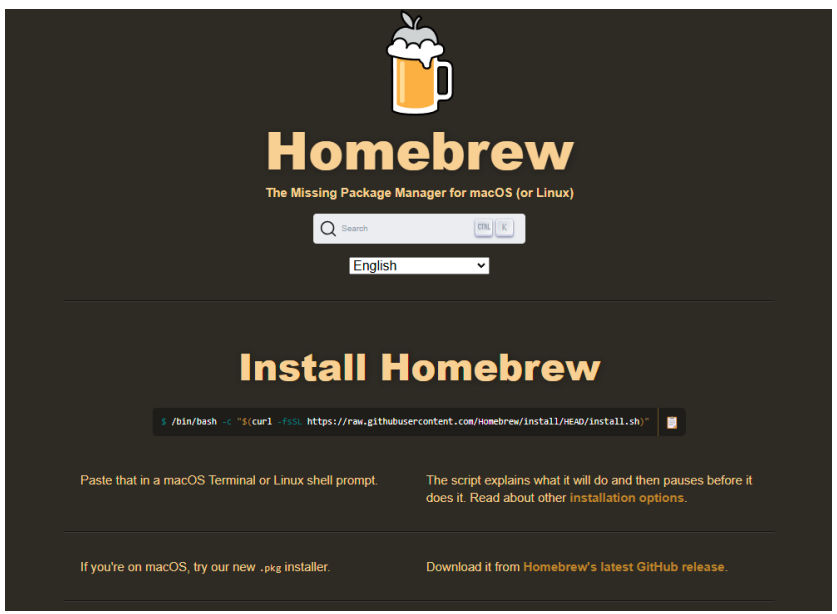
- In order to compile R for macOS, you will need **both** Xcode and GNU Fortran compiler.

Homebrew and GDAL for Mac users

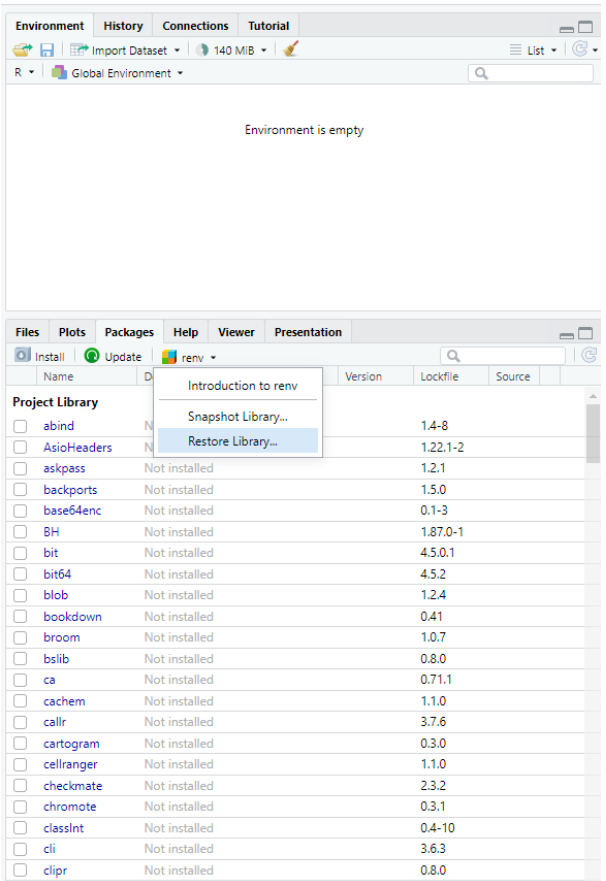
Next, use macOS Terminal to install Homebrew, and subsequently, GDAL.

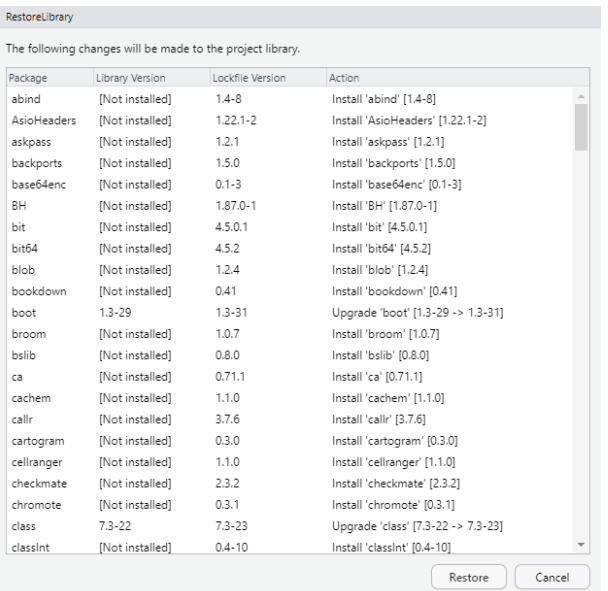
brew.sh

formulae.brew.sh/formula/gdal



Restoring the environment





- Navigate to the bottom right panel.
- Click the Packages tab, then the renv button, and finally select the `Restore Library` option.
- Click the `Restore` button in the pop-up panel.

Install $\LaTeX$

$\LaTeX$ is a high-quality typesetting system. While it may seem as a language, understanding it isn't necessary for our purposes. We will use it to export `Rmd` files with results into PDF files.

If you already use $\LaTeX$ and have it installed through `MiKTeX` or `TeX Live`, you can skip this step.

If you are unfamiliar with $\LaTeX$ and don't have it installed yet, simply run the following R code in the console to install it:

```
1 tinytex::install_tinytex()
```

Finish setup of the R environment

After completing all the steps above, you should now have a fully functional R environment.

You should keep the folder `Applied-Spatial-Statistics` on your computer.

- You may rename it to anything you like, but make sure you know where it is located (its directory path).
- Starting next week, you will be working inside this folder.
- All lab activity code files (`Rmd` files) will be stored here.
  - This is **mandatory** because the `renv` package ties the environment (R packages) to this project folder.
    - Similar to Python virtual environments, it is not a system-wide setup (for R packages).
  - If you remove the folder or try to run the code outside of it, you will need to restore the environment again.

References

- https://r-pkgs.org/
- https://book.the-turing-way.org/reproducible-research/renv.html
- https://rstudio.github.io/renv/articles/renv.html