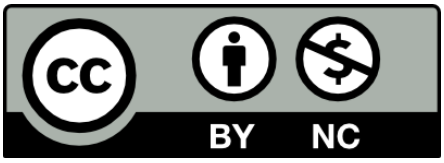
Getting Started with Visual Studio Code on macOS

Wooyong Park

Last Updated: May 15, 2025*



Contents

*The most updated version of this document is available here

1 VS Code

1.1 Why VSCode?

Visual Studio Code (VS Code) is a lightweight, versatile code editor developed by Microsoft. It supports a wide range of languages(R, Python, Julia, and so on) and features such as IntelliSense, version control integration, debugger, and extensibility through extensions. Compared to traditional IDEs, VSCode offers more control, speed, and flexibility for developers and data scientists alike.

But why use VSCode over other IDEs? There are great IDEs like Pycharm or Jupyterlab, and AI intensive IDEs like Zed or Cursor. The biggest reason I prefer VSCode is the value for money. GitHub Copilot runs on VSCode and is free for students and teachers. At the same time, it supports multiple langauges.

1.2 Installation

1.2.1 Via Web Browser

1. Visit https://code.visualstudio.com/ and download the macOS version.
2. Open the downloaded `.zip` file and move `Visual Studio Code.app` to `/Applications`.
3. Launch it from Launchpad or Spotlight.

1.2.2 Via Homebrew

1. Open Terminal.
2. Run the following command:

```
brew install --cask visual-studio-code
```

3. Launch it from Launchpad or Spotlight.

1.2.3 Should I use Homebrew?

My answer is yes. It just makes it so easy to install and uninstall softwares, it takes care of dependencies and helps **version control**! In my personal experience, I have never had any version conflict issues with R, but had multiple issues with Python. (e.g., Tensorflow is supported fully on Python 3.10, but not on 3.12.) FYI: If you want to maintain multiple versions of Python, I recommend using simplistic package manager like homebrew and install `pyenv` from it.

To install homebrew, run the following command in your terminal:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/
    HEAD/install.sh)"
```

1.3 Useful Extensions

VSCode relies heavily on extensions to enhance its functionality. Even if you have installed softwares like R or LaTeX, you need to install extensions to use them in VSCode. Here are some recommended extensions:

Click on the Extensions icon (or press `Cmd+Shift+X`) and search for:

- **Python** - Microsoft (probably pre-installed)
- **R** - REditorSupport
- **Jupyter** - Microsoft
- **Data Wrangler** - Microsoft
- **GitHub Copilot** - GitHub
- **GitHub Copilot Chat** - GitHub
- **LaTeX** - Mathematic Inc.
- **LaTeX Workshop** - James Yu

1.4 Easy Relative Pathing

VSCode allows you to open files and folders using relative paths. This is equivalent to maintaining RStudio's `.Rproj` in projects, which enables you to open RStudio in the same working directory as the project. An easy way to do this is to install `code` command in your PATH:

1. Open VSCode and press `Cmd+Shift+P`.
2. Search for `Shell Command: Install 'code' command in PATH`.
3. After the install, open your Terminal in the project directory. You can now type:

```
code .
```

   This opens the current folder in VSCode.

1.4.1 If the installation does not work

You can manually add the `code` command to your PATH through terminal(**Caution**: please check which shell you are using prior to running the following command):

- For `bash`:

```
nano ~/.bash_profile
export PATH="$PATH:/Applications/Visual Studio Code.app/Contents/Resources/app
    /bin"
source ~/.bash_profile
```

- For `zsh`:

```
    nano ~/.zshrc
    export PATH="$PATH:/Applications/Visual Studio Code.app/Contents/Resources
        /app/bin"
    source ~/.zshrc
```

1.5 Modifying Keyboard Shortcuts (with JSON)

1. Press `Cmd+Shift+P` → `Preferences: Open Keyboard Shortcuts (JSON)`
2. Example custom bindings(setting pipeline operator `%>%` in R):

```
[
  {
    "key": "cmd+shift+m",
    "command": "type",
    "args": { "text": " %>% " },
    "when": "editorTextFocus && editorLangId == 'r'"
  }
]
```

1.6 Miscellaneous

1.6.1 In Settings

Go to `Cmd + ,` (Settings) and explore:

- Adjust the theme: Search for `Preferences: Color Theme` in the Command Palette (`Cmd+Shift+P`) and choose a theme you like.
- Enable autosave: Go to Settings (`Cmd+,`) and search for `files.autoSave`. Set it to `afterDelay` or `onFocusChange`.
- Change font size: Search for `editor.fontSize` in Settings and adjust the value.
- Enable minimap: Search for `editor.minimap.enabled` in Settings and toggle it on or off.
- Customize tab size: Search for `editor.tabSize` in Settings and set your preferred number of spaces.
- Enable word wrap: Search for `editor.wordWrap` in Settings and set it to `on`.
- Format on save: Search for `editor.formatOnSave` in Settings and enable it.
- Change line height: Search for `editor.lineHeight` in Settings to adjust spacing between lines.
- Configure IntelliSense: Search for `editor.quickSuggestions` in Settings to enable or disable inline suggestions.

1.6.2 In Command Palette

Go to `Cmd + Shift + p` (Command Palette) and explore: The Command Palette (`Cmd+Shift+P`) is a powerful tool in VSCode. Here are some tips to make the most of it:

- Use keywords: Start typing what you want to do (e.g., "format", "install", "debug") to quickly find relevant commands.
- Access settings: Search for `Preferences: Open Settings (JSON)` or `Preferences: Open Settings (UI)` to customize your environment.

- Manage extensions: Search for `Extensions: Install Extensions` to browse and install new extensions.
- Run tasks: Use `Tasks: Run Task` to execute predefined tasks like building or testing your project.
- Search shortcuts: Look for `Preferences: Open Keyboard Shortcuts` to view or modify key bindings.
- Open files: Type `>Open File` to quickly locate and open files in your workspace.
- Git commands: Use commands like `Git: Commit` or `Git: Push` to manage version control directly from the palette.
- Debugging: Search for `Debug: Start Debugging` to launch the debugger.

2 Copilot in VSCode

2.1 Introduction

GitHub Copilot Pro is an AI-powered code assistant that

- suggests code completions
- generates code snippets
- is empowered by the newest versions of ChatGPT, Claude, and Gemini.

Some people prefer Cursor over Copilot, but I stick with Copilot because it is free for students and teachers. For comparisons of the two, check out this video.

2.2 GitHub Education Free Access

If you're a student or teacher, go to `https://education.github.com` and apply to get Copilot Pro for free.

2.3 Setup

1. Install the **GitHub Copilot** extension.
2. Sign in with your GitHub account.

2.4 Features

- Inline code suggestions (type `//` and a prompt).
- **Copilot Chat:** Press `Cmd+I` or use `Cmd+Shift+P → Copilot: Open Chat`.
- Copilot can open in a sidebar or a separate window.(`Cmd+Shift+I`)

3 LaTeXin VSCode

3.1 Setting up

1. Install LaTeX(MacTeX).
   (a) Download MacTeX from https://tug.org/mactex/
   (b) Install and ensure it's added to PATH (usually `/Library/TeX/texbin`)
2. Install **LaTeX** and **LaTeX Workshop** extension in VSCode.

3.2 Pane View

In VSCode, you can split the editor into multiple panes based on your preference. Unlike Texstudio where the pane is fixed, you can adjust the size of each pane in VSCode. Just drag the files you want to open into the editor in the desired place. For example,

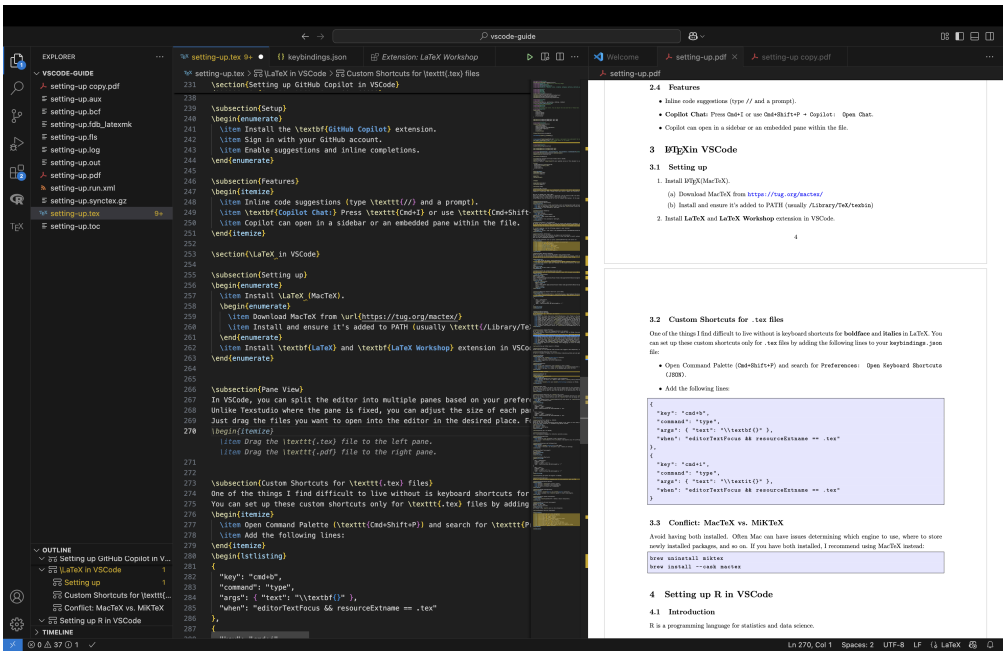


Figure 3.1: Left: sidebar, middle: .tex, right: .pdf

3.3 Custom Shortcuts for `.tex` files

One of the things I find difficult to live without is keyboard shortcuts for **boldface** and **italics** in LaTeX. You can set up these custom shortcuts only for `.tex` files by adding the following lines to your `keybindings.json` file:

- Open Command Palette (`Cmd+Shift+P`) and search for `Preferences: Open Keyboard Shortcuts (JSON)`.
- Add the following lines:

```
{
  "key": "cmd+b",
  "command": "type",
  "args": { "text": "\\textbf{}" },
  "when": "editorTextFocus && resourceExtname == .tex"
},
{
  "key": "cmd+i",
  "command": "type",
  "args": { "text": "\\textit{}" },
  "when": "editorTextFocus && resourceExtname == .tex"
}
```

3.4 Conflict: MacTeX vs. MiKTeX

Avoid having both installed. Often Mac can have issues determining which engine to use, where to store newly installed packages, and so on. If you have both installed, I recommend using MacTeX instead:

```
brew uninstall miktex
brew install --cask mactex
```

4 R in VSCode

4.1 Introduction

R is a programming language for statistics and data science. In my own preference, I prefer using R in RStudio over VSCode because of the familiarity. I found a lot of people who are used to VSCode because of other languages like Python or Julia still prefer using VSCode for R as well. But still, some people prefer using R in VSCode because of the **GitHub Copilot** and **Copilot Chat** features. Another alternative is the Positron IDE, but it is in its early stage of development and does not support Copilot yet.

4.2 Prerequisites

- Install R from https://cran.r-project.org/
- Install `XQuartz` from https://www.xquartz.org/ for better compatibility.

4.3 Extensions and Configuration

- Install **R Extension** by REditorSupport.
- After the installation, install **languageserver** package in R:

  ```
  install.packages("languageserver")
  ```

- The following softwares are recommended by the offical REditorSupport:

- **radian**: a terminal-based R console. [Install]
- **R-Debugger**: a debugger for R.
- **httpgd**: a package for graphics device. Needed for plotting in VSCode. [Install]

- Enable R Terminal: set the path to your R binary in settings.

4.4 Check Environment

In R terminal:

```
sessionInfo()
```

4.5 Unsaving/Unrestoring .RData

In RStudio, you might have had the following global settings:

- Tools - Global Options - General: Uncheck restore .RData into workspace at start up.
- Tools - Global Options - General: Choose 'never' for Save workspace to .RData on exiting.

These settings prevent R from restoring the workspace when you open the project again, so that you can start with a clean slate every time you open the project.

In VS Code, these settings are not really necessary because VS always launches a fresh R terminal when you open a project. However, if you explicitly run `q()` or use R interactively, you might want to set these settings in VSCode as well. It will help you avoid surprises from `.RData` files that are automatically loaded when you open a project, and is also helpful for reproducibility and consistency across different environments.

- Open Command Palette (`Cmd+Shift+P`) and search for `Preferences: Open Settings (JSON)`.
- Add the following lines:

```
{
    "r.rterm.option": [
        "--no-save",
        "--no-restore-data"
    ]
}
```

4.6 Custom Shortcuts

I recommend setting up custom shortcuts for the `%>%` operator and the assignment operator `<-` in R. You can set up these custom shortcuts by adding the following lines to your `keybindings.json` file. Note that these shorcuts are set up for `.qmd`, `.rmd`, and `.r` files only. If you want to set up these shortcuts for the R console in terminal, you can add `terminalFocus` to the `when` condition.

```
{
  "key": "cmd+shift+m",
  "command": "type",
```

```
    "when": "editorTextFocus && editorLangId =~ /qmd|rmd|r/",
    "args": { "text": " %>% " }
},
{
    "key": "alt+-",
    "command": "type",
    "when": "editorTextFocus && editorLangId =~ /qmd|rmd|r/",
    "args": { "text": " <- " }
}
```

5 Python and Jupyter in VSCode

5.1 Introduction

Python3 is a general-purpose language great for data analysis, web, and ML.

5.2 Installation Options

- Homebrew: `brew install python3`
- Anaconda: https://www.anaconda.com
- Use built-in Python (not recommended)

5.3 VSCode Configuration

- Install **Python** and **Jupyter** extensions.
- Open a notebook file (`.ipynb`) to start using Jupyter.
- (Optional but recommended) Create a virtual environment for your project(see 5.4)
- Once the python interpreter is selected, run:

```
python3 -m pip install notebook jupyterlab
```

- If you are using a virtual environment, also run:

```
    python3 -m pip install ipykernel
```

5.4 Creating a Virtual Environment

5.4.1 error: externally-managed-environment

If you have installed Python via Homebrew, you may encounter `error: externally-managed-environment` when trying to install packages. This is because Homebrew's Python is managed by the system, and since 2023, Homebrew's version of Python has been blocking the *global* installation of packages by adding a file called `EXTERNALLY-MANAGED` in your Python directory.

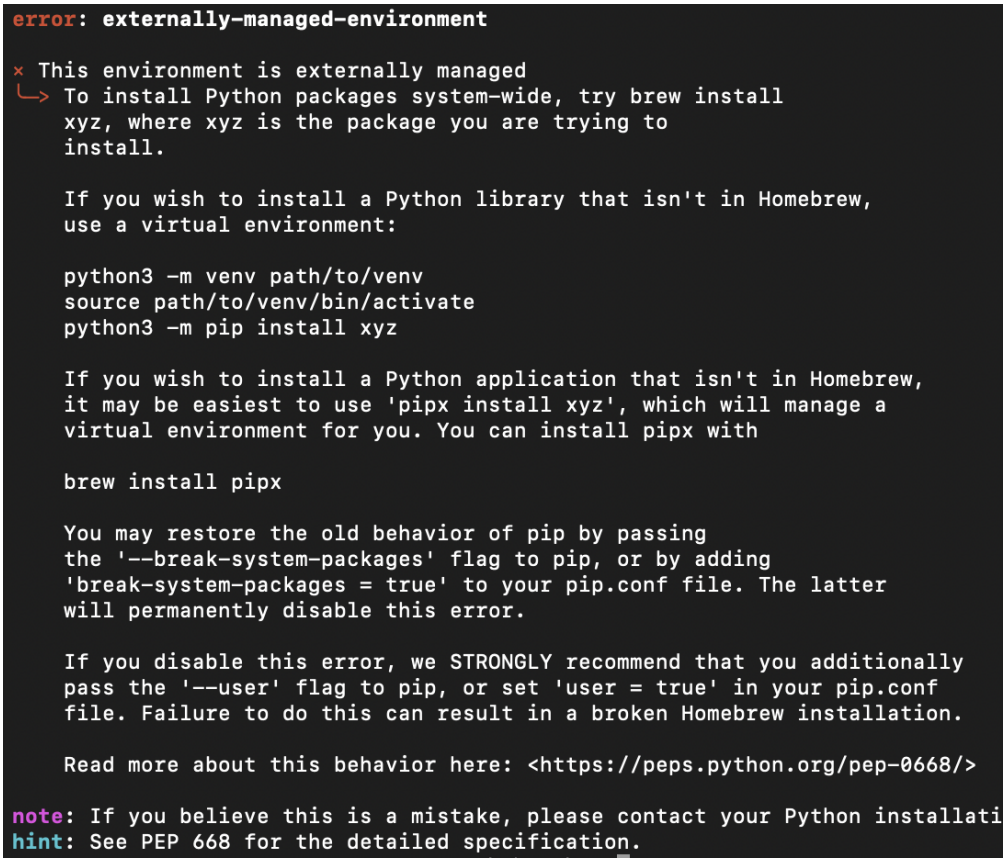


Figure 5.1: Error: externally-managed-environment

The most straightforward fix is creating a virtual environment. At first sight, it may seem like a hassle, but it is actually a good practice both for your PC and reproducibility of your codes. The concept of virtual environments is similar to R's `renv` package, which creates a virtual environment for your R project: you maintain different versions of packages for different projects, so that even if a package is updated or downgraded globally, your project will not be affected by the update. **Fortunately, VSCode helps you create these virtual environments easily.**

5.4.2 Creating .venv

- Through terminal:

```
python3 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
```

- Through VSCode:
  - Open Command Palette (`Cmd+Shift+P`) and search for `Python: Select Interpreter`.
  - Select `+ Create Virtual Environment`.
  - Choose the environment type (e.g., `venv`).
  - Select the Python interpreter. (If you are using homebrew, select the one in `/opt/homebrew/bin/python3`.)
  - Wait for the environment to be created.
  - Activate the environment by running `source .venv/bin/activate` in the terminal.
  - Install packages using `pip install -r requirements.txt` or through the Command Palette.
  - If you have a `requirements.txt` file, you can install all the packages listed in

Caveat: Sometimes, you might want to have a different name for your virtual environment rather than `.venv`. VSCode names your environment as `.venv` by default. In that case, simply renaming the folder `.venv` won't work because the packages inside that folders still use the name `.venv`. What I do to have a different name from it is simply to use the terminal to create one with a different name. Keep things simple and get used to using the terminal.

5.5 Select Python Interpreter

- Press `Cmd+Shift+P → Python: Select Interpreter`
- Set VSCode to use `.venv` as the interpreter.
- As mentioned above, you have to install

```
python3 -m pip install notebook jupyterlab
python3 -m pip install ipykernel  #if using venv
```

6 Keyboard Shortcut Cheatsheet

- `Cmd+P` - Quick file navigation
- `Cmd+Shift+P` - Command Palette
- `Cmd+‘` - Toggle terminal
- `Cmd+B` - Toggle sidebar
- `Cmd+K Cmd+S` - Keyboard shortcut viewer
- `Cmd+/` - Toggle line comment
- `Cmd+Shift+F` - Find in project
- `Cmd+Alt+Arrow` - Multi-cursor editing
- `Cmd+I` - Open Copilot Chat inside the script
- `Cmd+Shift+I` - Open Copilot Chat in a separate window
- `Ctrl+‘` - Open terminal