# **Setting Up R/RStudio for New Instructors**

AY 2025-2026

This serves as an RStudio guide for newcomers to the 141/142 sequence.

In AY ‘25/’26 the core curriculum is using posit.cloud for R. This is a cloud platform that can be accessed from any computer. Cadets and instructors will all have accounts on posit.cloud; the expectation is that posit.cloud is our primary tool for R. Students will have an *option* to install R/RStudio to their local machines, purely as a backup option.

If you are slated to teach Math 141/142 or Math 300 in Fall ‘25, then you should already have a posit.cloud account. Don’t pay for your own account, wait to be invited by an admin so that you can use the USAFA license. While you have a perfectly great posit.cloud account, be sure to install R/RStudio on you laptop so that you have a backup option in case Wi-Fi, internet, or posit.cloud go down – those are all real possibilities.

If you are not slated to teach Math 141/142 or Math 300 in Fall ’25, then you won’t have a license available to you. In that case, you’ll need to pay for your own subscription (not recommended) or install R to your laptop (that’s the preferred option).

Please report any issues you encounter to Joe Eichholz.

# Posit.Cloud

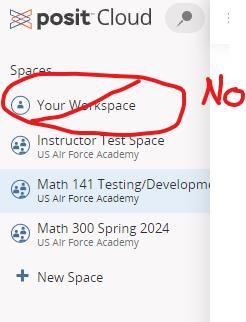
1. Accept the invitation to posit.cloud as a whole. You'll need to set a password and verify your email address. This is just getting you an account associated with USAFA so that you don’t have to pay for compute time. This is the account you’ll use to join the workspace.
2. Accept the invitation to the Summer 2025 NIT workspace.

Posit.cloud has two organizational layers. First is the workspace, you can be members of one or many workspaces, with various levels of access to each workspace. Within workspaces are projects, which are normal R projects. You can think of them as folders that hold your files and data. We can control access to individual projects as well.

1. Once you accept the invitation to the workspace, you should see it in the menu on the left-hand side of the screen, like so:   
     
   A screenshot of a computer

   AI-generated content may be incorrect.

1. Click on the workspace. **Don't do anything in your personal workspace!** Your personal workspace is not part of the USAFA account, so USAFA doesn't pay for the compute time you use in that workspace. Eventually you'll run out of compute time on your personal workspace and then posit.cloud won't let you do anything more.



1. Once you click on the Summer 2025 NIT workspace, you'll see all of the projects that are a part of the workspace. Since you are a moderator you can see all of the projects in the workspace; students will only be able to see their own projects. **Create your own copy** of the “Base Project”. Since you are a moderator, you can make changes to the base project that would impact everyone else. Instead, make a copy of the “Base” project that only you will work in. Student’s wont have this ability. There is only one button they can push, labeled “start”, and that will automatically create their own private copy of the project.

A black and white screen

AI-generated content may be incorrect.

1. Now change the title of your project to something like: <your last name> testing or whatever. Just make it distinguishable from other projects.   
     
   A screenshot of a computer

   AI-generated content may be incorrect.

1. That’s it. Now whenever you want to use RStudio you just log in to posit.cloud, open your project, and work away.

# Install on Laptop

To install locally, follow these directions. If you run in to issues, please let Joe Eichholz know.

**Download R/RStudio from the self-service client**

1. Open the Self-Service Client either from the desktop or start menu.

A red circle with a hand pointing at a screen

Description automatically generated

1. In the upper right-hand corner, search for “R bundle”. You should find an option to install R bundle – latest. Install that. *It will take a while to install.*

**A blue sky with white clouds

Description automatically generated**

1. You can check on the progress of your install in the “Activity” tab on the left-hand side.

**A close-up of a computer screen

Description automatically generated**

1. Once installed, open RStudio. Just type “RStudio” in the search bar.   
   **A screenshot of a computer

   Description automatically generated**
2. When asked, keep RStudio using “machines’ default 64-bit version of R”  
     
   **A screenshot of a computer

   Description automatically generated**

**Install Python**

Some of our unique packages rely on Python. Install that from the self-service center.

1. Search for Python. You’ll find a few matching options.

A blue and white background

Description automatically generated

1. Install the latest version of Python, not an IDE. In this example that is Python 3.11.4. *This will take a while to install.*

**Install packages unique to Calc/USAFA**Great, now you have a basic R/RStudio and Python setup. Time to install the packages that we use in 141 and 142. To do this, download the file SetupUSAFACalc.R. Open the file in RStudio using File 🡪 Open File … The file will open and your console will look like this:  **A screenshot of a computer

Description automatically generated**

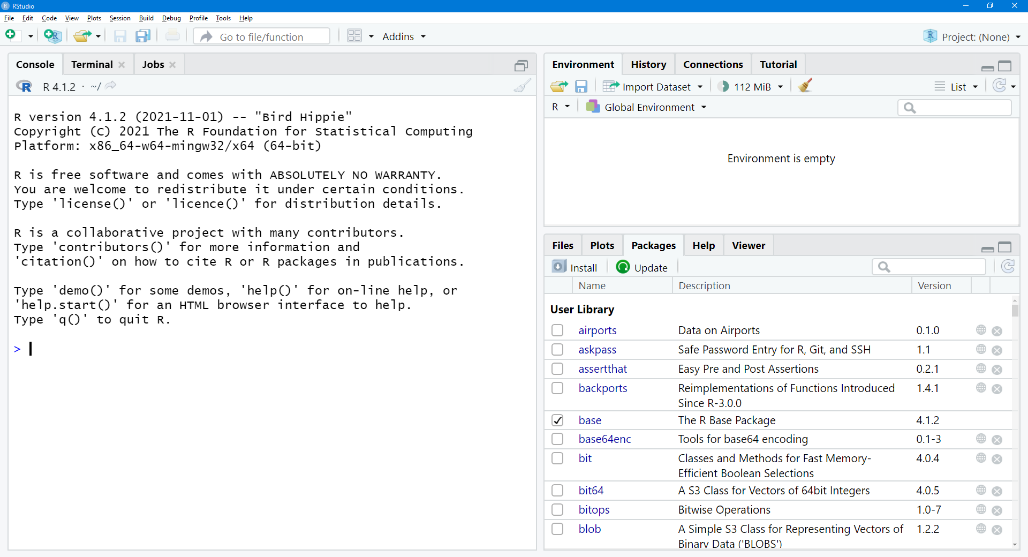
1. Click the Source button on the top-right of the script window.  **A screenshot of a computer program

   Description automatically generated**
2. You will be asked if you would like to create a default Python environment for the reticulate package. Click yes.   
     
   **A computer screen shot of a computer error

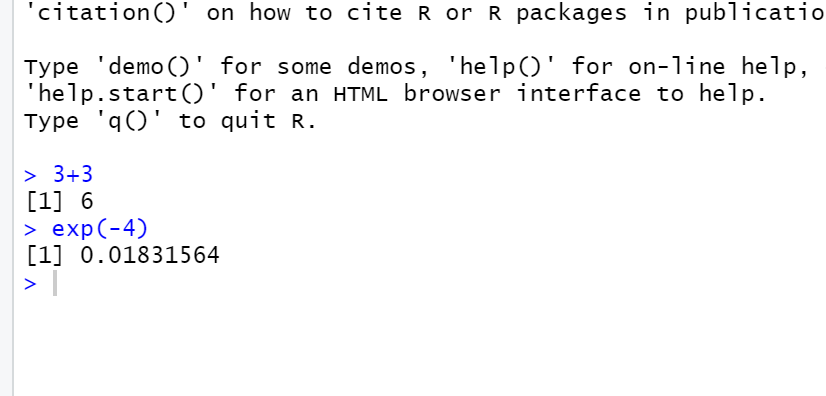
   Description automatically generated**
3. If R says that it should be restarted before installation of package X, click OK.
4. If R asks if you want to update packages, you can either say yes or no. In theory it does not matter. Now might be a good time to get everything updated, though.
5. Assuming all went well, close RStudio and open it up again.

# Welcome

Welcome to RStudio. Your first time opening RStudio, you’ll see three boxes.



The large box on the left is your “console”. This serves as a calculator. You can type simple commands like 3+3 or exp(-4) (for ).

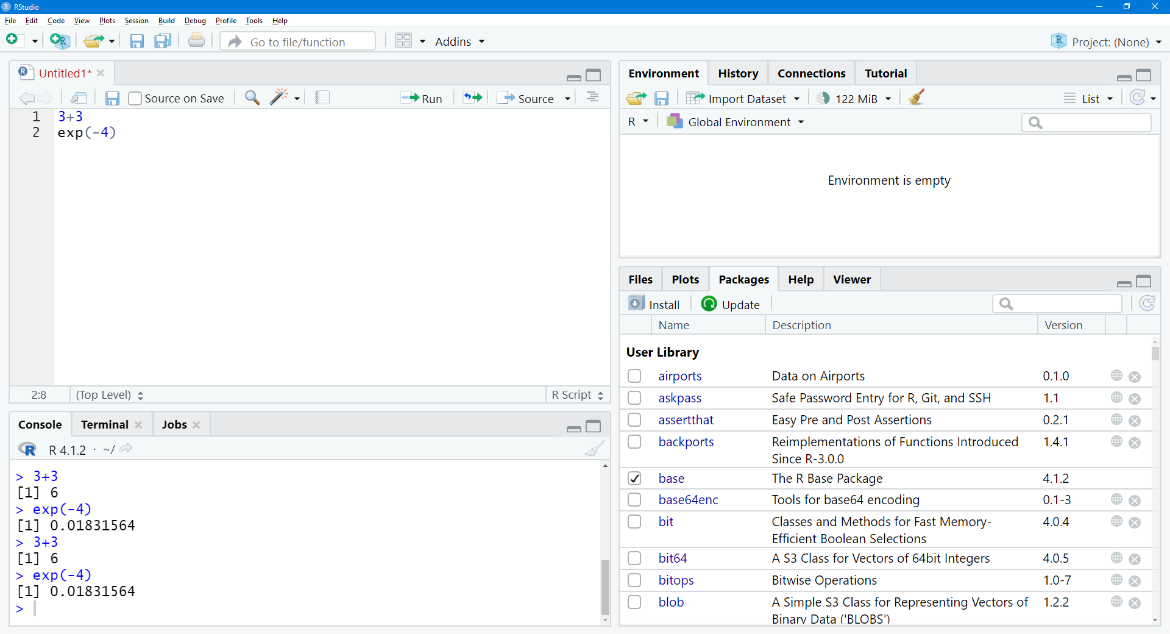


The box on the top right shows your environment. Variables that you have declared will appear here. The box on the bottom right has several tabs, and we will use this box for displaying plots, accessing help, and viewing our list of packages.

You can also work in scripts in R. To explore a script, click File -> New File -> R Script. This will open up a blank document. You can type commands here like 3+3 or exp(-4). There are two ways (that I can think of) to run commands in an R script file.

1. Put your cursor on the line you want to run. Hit ctrl+r, the command on the line with the cursor will run. If you want to run multiple lines, highlight the desired lines and then hit ctrl+r.
2. Highlight one or more lines and then click the Run button in the upper right corner of the window.

You can save scripts to your computer. They will be files of type .R.



# R Basics

In the textbook, and in other getting started guides you may see around the department, you will see references to “loading packages” or “libraries” or “requiring libraries”, etc. Like any programming language, R has a package system that allows for extending the functionality of the language. The packages that we use extensively in the course are mosaicCalc, and MMAC. In the past, we started ever R session with “require(mosiacCalc)” and “require(MMAC)”. We’ve configured RStudio in such a way that ***this is no longer needed***; those packages (should) be loaded every time you start RStudio. If you run the commands from the book nothing will be hurt, but nothing will be gained. **Let me know if you run into a situation where this is not true.**

The R language has many applications, but in this sequence of courses, we will use it primarily for creating and plotting functions, fitting functions to data, obtaining derivatives and antiderivatives, and other calculus related applications.

Block 1 is all about defining, using and visualizing different families of functions. Therefore, in this block, the main two commands we will use are makeFun() and plotFun().

## Making a Function

makeFun(): creates a function; requires tilde expression. A ***tilde expression*** is the body of the function followed by a tilde and the input variable. For example, the tilde expression for is x^2 ~ x. This can be read as x^2 with variable x.

To use makeFun(), we input a tilde expression and assign the function to a name. In this case, since , we will call our function f. To assign to a variable we can use = or <-.

f = makeFun(x^2 ~ x)

f <- makeFun(x^2 ~ x)

CAUTION: ***R is case sensitive***. A very common mistake among new users is to use the wrong case of a variable, leading to an error.

Once created, note that the function appears in the global environment in the top right box. This function can now be evaluated/plotted. For example, we can type f(2) into the console or script to find the value of the function evaluated at .

## 

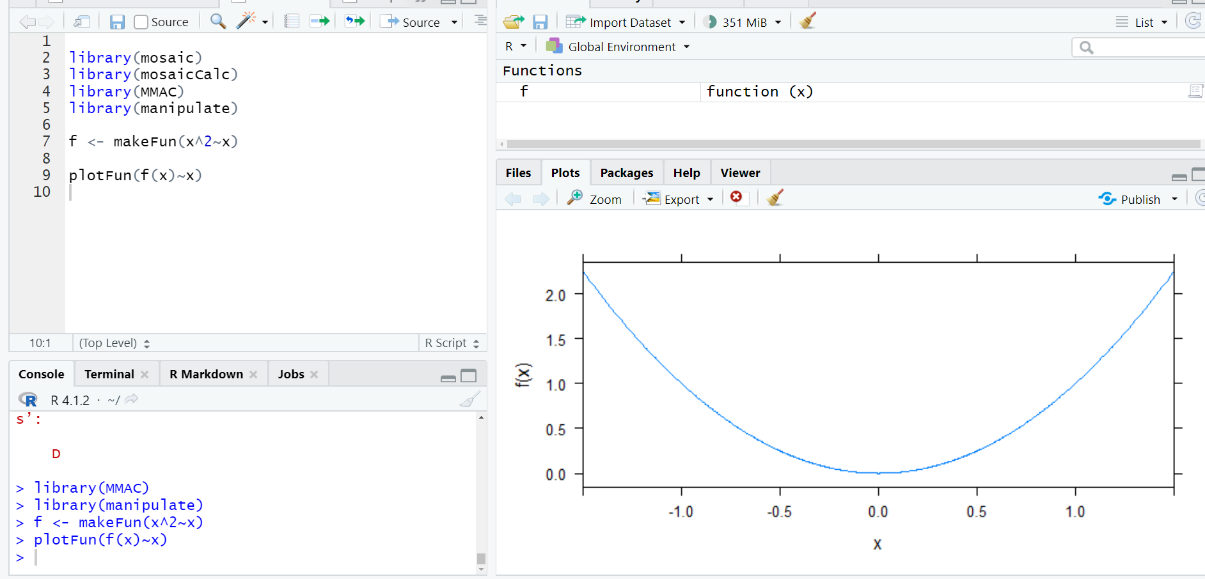
## Plotting a Function

plotFun(): Plots functions; also requires tilde expression. We can input a function defined by makeFun() or we can use the tilde expression directly. For example:

plotFun(x^2~x)

OR

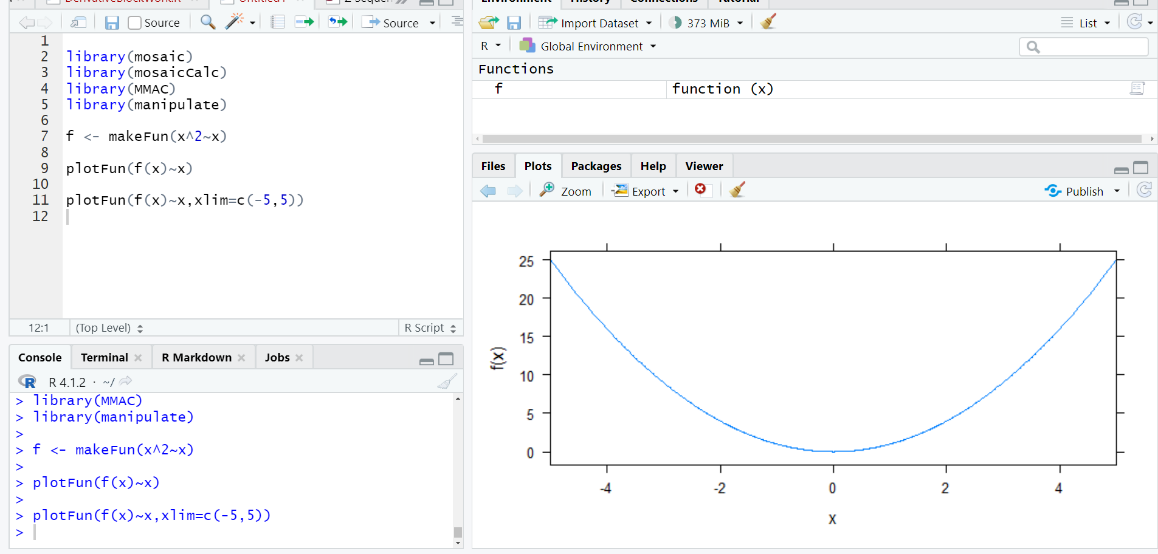
plotFun(f(x)~x))



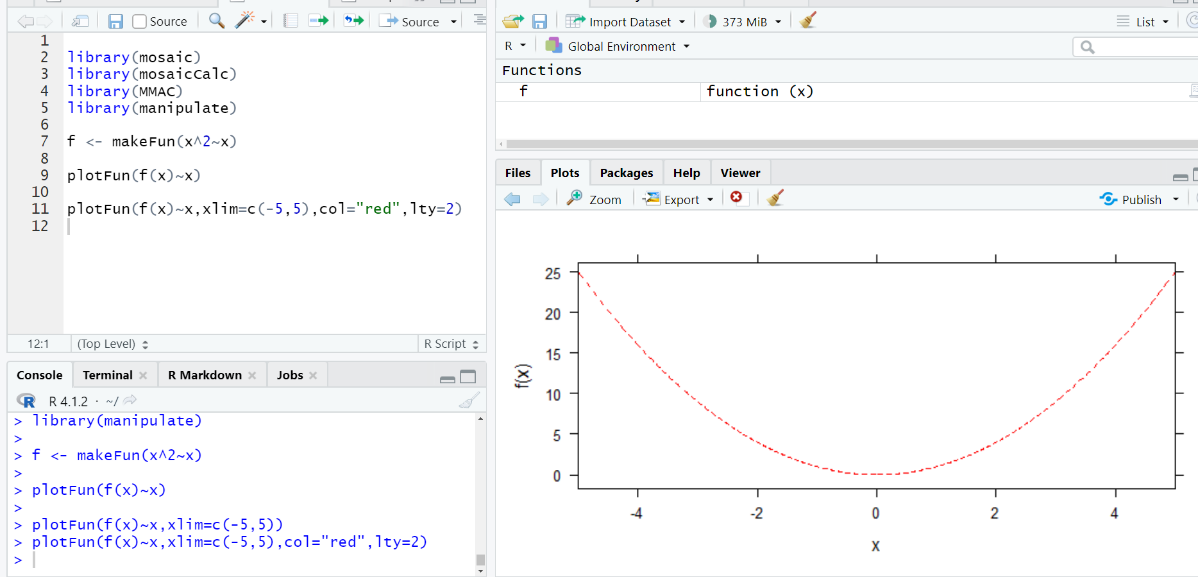
Note that I can use additional options with plotFun(). For example, I could adjust the visible graphics window by using xlim (which controls the horizontal axis) and ylim (which controls the vertical axis):

plotFun(f(x) ~ x, xlim = c(-5,5))

NOTATION NOTE: The xlim argument expects two values: a min and a max. To give xlim both of these values simultaneously, we use the c() function, which combines multiple values into a vector.



We could also change the appearance of the plot in other ways. For example, color and line type are controlled by the col and lty arguments.



Finally, we can add multiple plots onto the same set of axes by adding the add=T argument:

