# RStudio Basic Usage

**Updated 2025.10.27**

## A. The interface

The RStudio interface has 4 main windows that are preconfigured out of the box. Before you create or open a file, there will be only 3 showing. The left side will show the Console pane, the top right will be the Environment pane, and bottom right will be the Directory pane. Once you open or create a file, it will open on the top left (Source pane) and the console will be on the bottom left.

### Source Pane

This is the R/Rmd file you will be working in. This is where you primarily work with R.

### Console Pane

This is where base R shows up. If you were to work with R without an IDE, this is what it would be like. You can still interact with R directly here. Whatever run in the R console will run and render to the environment but it will not be reflected in the Source Pane. Thus, if you make changes to an object here, it will not be recorded in the script.

### Environment Pane

The Environment Pane shows what is included in your environment. If a dataset is imported in as an object, it shows up here. You can click on the things listed to see more information about them.

### Directory Pane

The default tab of the Directory Pane is the Files tab which simply shows the contents of your directory. You can also move around into other directories here. This pane also contains many other tabs:

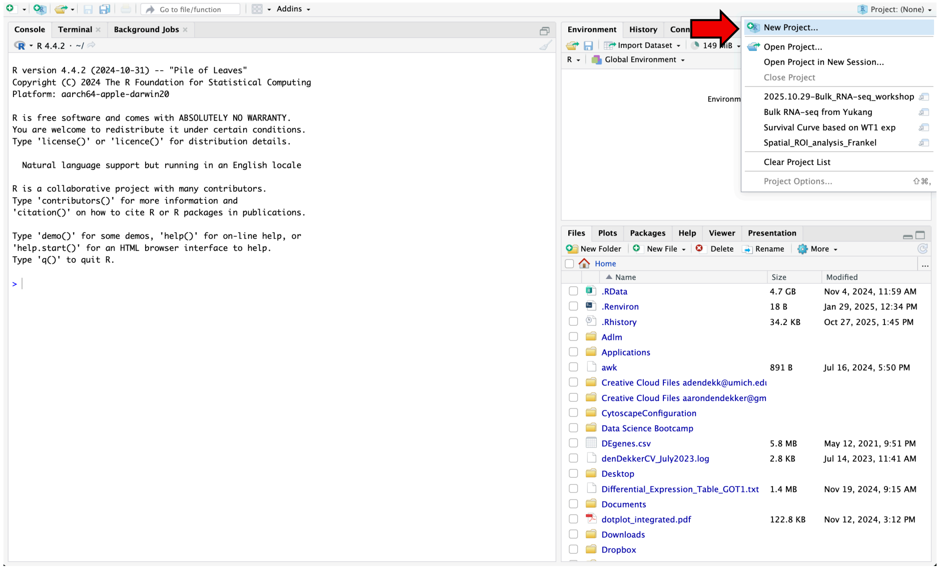
- Plots – which will render plots run in an R Script. If plots are run in a Markdown, they appear in the Source pane below the chunk that generated them.
- Packages – shows the R packages that are available and which ones are loaded.
- Help – you can search for documentation to any R package here.

The layout can be configured by selecting **Panes** in the **View** tab in the RStudio banner.

## B. Creating a new project

1. Open up RStudio.
2. In the top right of the interface, select the **Project (None)** dropdown and select **New Project…**

(if you are already working in project, the dropdown tab will be title with the name of the project you're working in. If you want to start a new project, you'll have to close the project in the dropdown menu.)



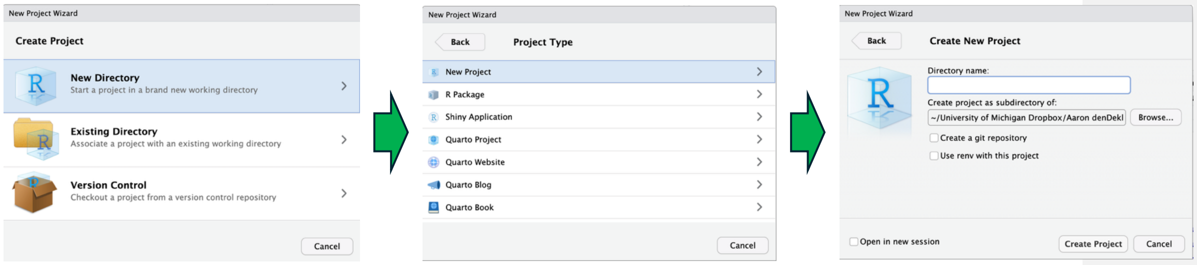
```
R version 4.4.2 (2024-10-31) -- "Pile of Leaves"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: aarch64-apple-darwin20

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 
```

| Name | Size | Modified |
|---|---|---|
| .RData | 4.7 GB | Nov 4, 2024, 11:59 AM |
| .Renviron | 18 B | Jan 29, 2025, 12:34 PM |
| .Rhistory | 34.2 KB | Oct 27, 2025, 1:45 PM |
| Adlm | | |
| Applications | | |
| awk | 891 B | Jul 16, 2024, 5:50 PM |
| Creative Cloud Files adendekk@umich.edı | | |
| Creative Cloud Files aarondendekker@gm | | |
| CytoscapeConfiguration | | |
| Data Science Bootcamp | | |
| DEgenes.csv | 5.8 MB | May 12, 2021, 9:51 PM |
| denDekkerCV_July2023.log | 2.8 KB | Jul 14, 2023, 11:41 AM |
| Desktop | | |
| Differential_Expression_Table_GOT1.txt | 1.4 MB | Nov 19, 2024, 9:15 AM |
| Documents | | |
| dotplot_integrated.pdf | 122.8 KB | Nov 12, 2024, 3:12 PM |
| Downloads | | |
| Dropbox | | |

**3.** In the “New Project Wizard” box that opens, select **New Directory**

**4.** Then select **New Project**

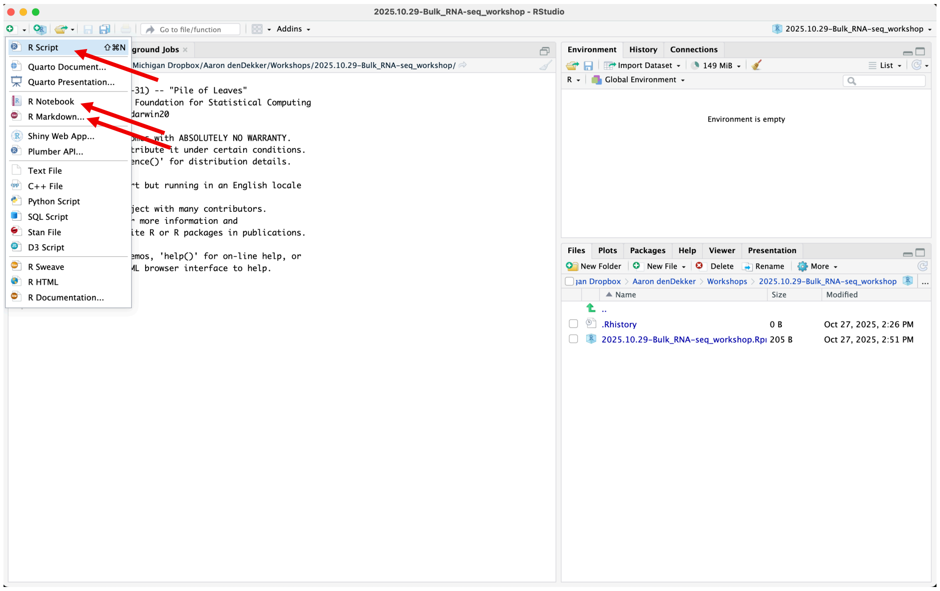
**5.** Use the “browse” button to move to the main directory want to work in and provide a name for the new directory in the “Directory name:” box.



****NOTE:** If you already have a directory where you have your data and that is where you want to work from, select **Existing Directory** and then use the browse button to select the directory.

## C. Creating a new file

In order to work effectively work in RStudio, you will need to create a new working file or open an existing one. You can either select **New File** from the **File** tab at the top or select the ⊕ ▾ button at the top left. Select either **R Script**, **R Notebook**, or **R Markdown...**



```
Michigan Dropbox/Aaron denDekker/Workshops/2025.10.29-Bulk_RNA-seq_workshop/

-31) -- "Pile of Leaves"
 Foundation for Statistical Computing
darwin20

omes with ABSOLUTELY NO WARRANTY.
tribute it under certain conditions.
ence()' for distribution details.

rt but running in an English locale

ject with many contributors.
r more information and
ite R or R packages in publications.

emos, 'help()' for on-line help, or
ML browser interface to help.
```

Environment is empty

ɟan Dropbox > Aaron denDekker > Workshops > 2025.10.29-Bulk_RNA-seq_workshop

| Name | Size | Modified |
|---|---|---|
| .. | | |
| .Rhistory | 0 B | Oct 27, 2025, 2:26 PM |
| 2025.10.29-Bulk_RNA-seq_workshop.Rpı | 205 B | Oct 27, 2025, 2:51 PM |

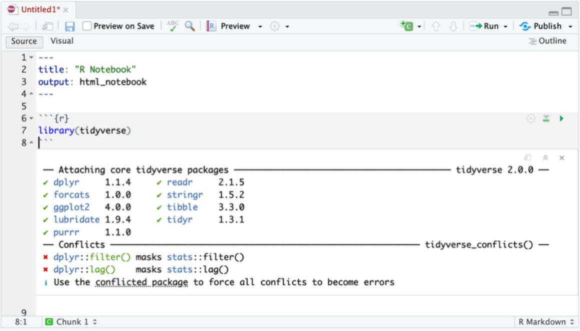
## R Script vs. R Notebook vs. R Markdown

### R Script

The R Script format is plain text. It is meant to contain only executable code. Each line of code is run line by line unless subsequent lines are part of a function and properly formatted. Any non-code text, e.g., notes about a particular function, must be commented out by adding **#** in front of the text you don't want to be run as code. You can also use **#** to keep any line of code from being run when the script is executed. In other words, you can keep lines of code to go back to if need be, but they won't be run when the whole script is executed.

### R Notebook

R Notebook will automatically bring up a Markdown format with a header for the document title and designate the output as an R Notebook. R Notebook uses R Markdown language and all code must be put inside code "chunks". Code chunks are designated with ```{r} at the top of the chunk, then the code to be executed, then a ``` to close (see example figure). An R Notebook is in HTML format.

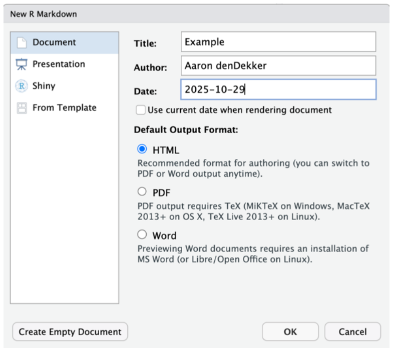


~~~
---
title: "R Notebook"
output: html_notebook
---

```{r}
library(tidyverse)
```
~~~

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✔ dplyr     1.1.4     ✔ readr     2.1.5
✔ forcats   1.0.0     ✔ stringr   1.5.2
✔ ggplot2   4.0.0     ✔ tibble    3.3.0
✔ lubridate 1.9.4     ✔ tidyr     1.3.1
✔ purrr     1.1.0
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package to force all conflicts to become errors
```

**R Markdown**

Selecting R Markdown will bring up a new panel to give a title to the file. The Author and Date fields will be filled with your name and the current date by default but those can be changed. You can also select type of output you want the report to be in: HTML, PDF, or Word. HTML is the default.



New R Markdown

Title: Example

Author: Aaron denDekker

Date: 2025-10-29

□ Use current date when rendering document

Default Output Format:

● HTML
Recommended format for authoring (you can switch to PDF or Word output anytime).

○ PDF
PDF output requires TeX (MiKTeX on Windows, MacTeX 2013+ on OS X, TeX Live 2013+ on Linux).

○ Word
Previewing Word documents requires an installation of MS Word (or Libre/Open Office on Linux).

# D. Using Markdowns

## 1. Selecting between R Markdown or R Notebook

Using R Markdown or R Notebook allows you to run your code in "chunks". If the output of the chunk generates a graphic or a table, the output is included in the report. Thus, it's a great way to keep a notebook of your analysis. Additionally, both R Markdown and R Notebook use Markdown language and allow you to add headers and create sections, create a table of contents, add notes, and emphasize parts of text. Although they are functionally the same, they act a little differently out of the box AND they work a bit differently under the hood.

- Both R Notebook and R Markdown have a header that contains information about your report. R Notebook brings up a blank header with fields for "title" and "output" which you can fill in. R Markdown brings up a panel that allows you to add a title, change the author name, and add a date. The information here populates the fields in the header once you select OK. You can always change what you have entered. Importantly, the header can be modified however you would like and additional fields can be added such "subtitle", "abstract", etc.

- When you select either R Notebook or R Markdown, a dropdown button is shown on the bar labeled named **Preview** or **Knit**, respectively. The **Knit** dropdown allows you to select HTML, PDF, or Word and once you select one, it will generate a report in that format. **Preview** works in a very similar way except that it also gives you a "Preview" option. The preview option will generate a HTML-formatted page but it will only generate code output for code that you have actually run. If you have code included in your script that you haven't run, it will print the code, but it won't print any output. Conversely, when you **Knit** and select the output format, all lines of code will be run, even if you haven't run them, and include the output in the report. This can have several implications:

    - If you haven't run all of the chunks, the Preview won't show what you haven't run.
    - If you generated a Notebook page with Preview but have changed the code, the Notebook page won't be updated unless you rerun Preview.
    - If you have memory expensive code chunks, such as importing a large data objects or running a differential expression analysis, Knit will rerun all of those processes. Thus, once you are done and want to generate a final document, Knit could take an extremely long time AND could fail since it will try to allocate the memory intensive processes to memory.

  **I suggest using R Notebook, making sure the output is "html_notebook" in the header, running through your analysis and pipeline iteratively and completely, and then generating your HTML report with Preview.**

## 2. R Markdown language

Markdown language is extremely useful for annotating your work. There are too many features to show here but below are some of the basic useful ones. For a more complete list of the features use this link.

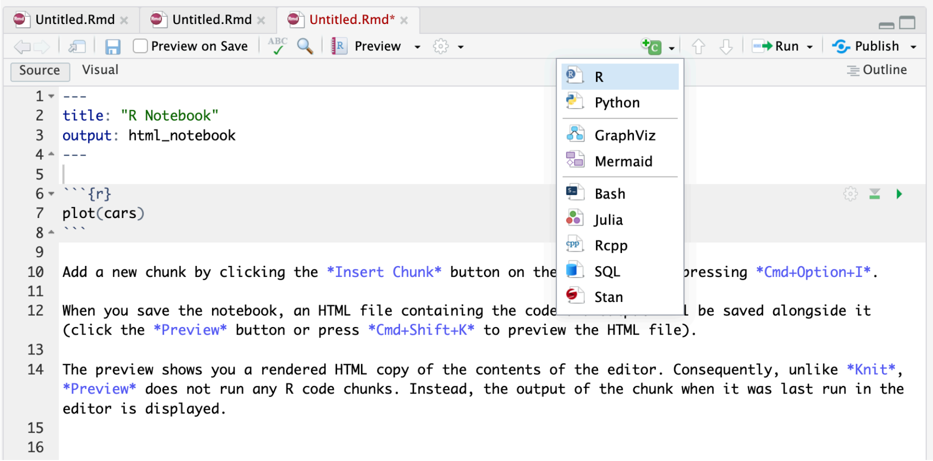
Basic Annotation

| | |
|---|---|
| # Header 1 | **Header 1** |
| ## Header 2 | **Header 2** |
| ... | |
| ###### Header 6 | **Header 6** |
| *italics* | *italics* |
| **bold** | **bold** |
| ***bold and italics*** | ***bold and italics*** |
| superscript^2^ | superscript$^2$ |
| subscript~2~ | subscript$_2$ |
| ~~strike through~~ | ~~strike through~~ |

Deleted: strike through

## 3. Creating code chunks

Code chunks must be separate from Markdown text. To create a code chunk, you can select the code chunk dropdown box and select the language you are working with.



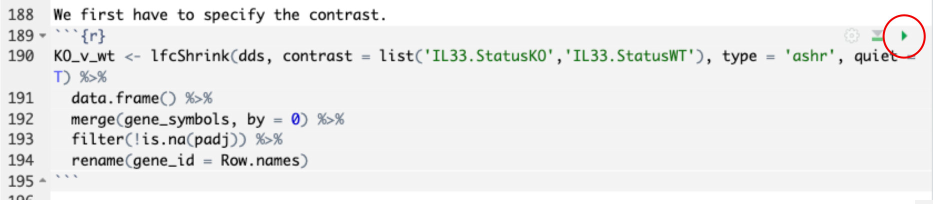
~~~
---
title: "R Notebook"
output: html_notebook
---

```{r}
plot(cars)
```

Add a new chunk by clicking the *Insert Chunk* button on the                    pressing *Cmd+Option+I*.

When you save the notebook, an HTML file containing the code                   l be saved alongside it
(click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*,
*Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the
editor is displayed.
~~~

Alternatively, and more conveniently, you can click option+command+I simulateneously on a Mac or Ctrl+Alt+I on Windows to quickly add a new code chunk.

Code chunks are delineated with a ```{r} at the top and ```. Thus, you can simply type ```{r} to enclose the top, then the code, and then ``` to enclose the bottom. Once, the code is enclosed, the

4. **Executing Code**

   Code chunks can be run by selecting the green arrow button in the code chunk box (see example figure).



~~~
We first have to specify the contrast.
```{r}
KO_v_wt <- lfcShrink(dds, contrast = list('IL33.StatusKO','IL33.StatusWT'), type = 'ashr', quiet = T) %>%
  data.frame() %>%
  merge(gene_symbols, by = 0) %>%
  filter(!is.na(padj)) %>%
  rename(gene_id = Row.names)
```
~~~

   Alternatively, code within a chunk can be run line by line but placing the cursor on the line you want to run and then typing command+return on a Mac or Ctrl+Return on Windows.

   When code is run in R Notebooks, if there is any output (plots, tables, etc.), these will show up in line just below the code chunk. If you run line by line, each time you run a line this will update what is represented. For example, if you have a chunk of code that has a line calculating variance stabilizing transformation and a separate line plotting a PCA from that, if you run the whole chunk, it will run the calculation and then plot the PCA under the code chunk. If you go back and run just the VST calculation line, the plot will disappear unless you rerun the line of code for plotting.