Introduction to R

In this course we will be using the R software environment for all our analysis. Throughout the course you will learn R and data analysis techniques simultaneously. However, we need to introduce basic R syntax to get you going. In this section, rather than cover every R skill you need, we introduce just enough so that you can follow along the remaining sections where we provide more in-depth coverage, building upon what you learn in this section. We find that we better retain R knowledge when we learn it to solve a specific problem.

In this section, as done throughout the course, we will use a motivating case study. We ask a specific question related to crime in the United States and provide a relevant dataset. Some basic R skills will permit us to answer the motivating question.

US gun murders

Imagine you live in Europe and are offered a job at a US company with many locations across all states. It is a great job but news with headlines such as **America is one of 6 countries that make up more than half of guns deaths worldwide** have you worried. Charts like this make you worry even more:

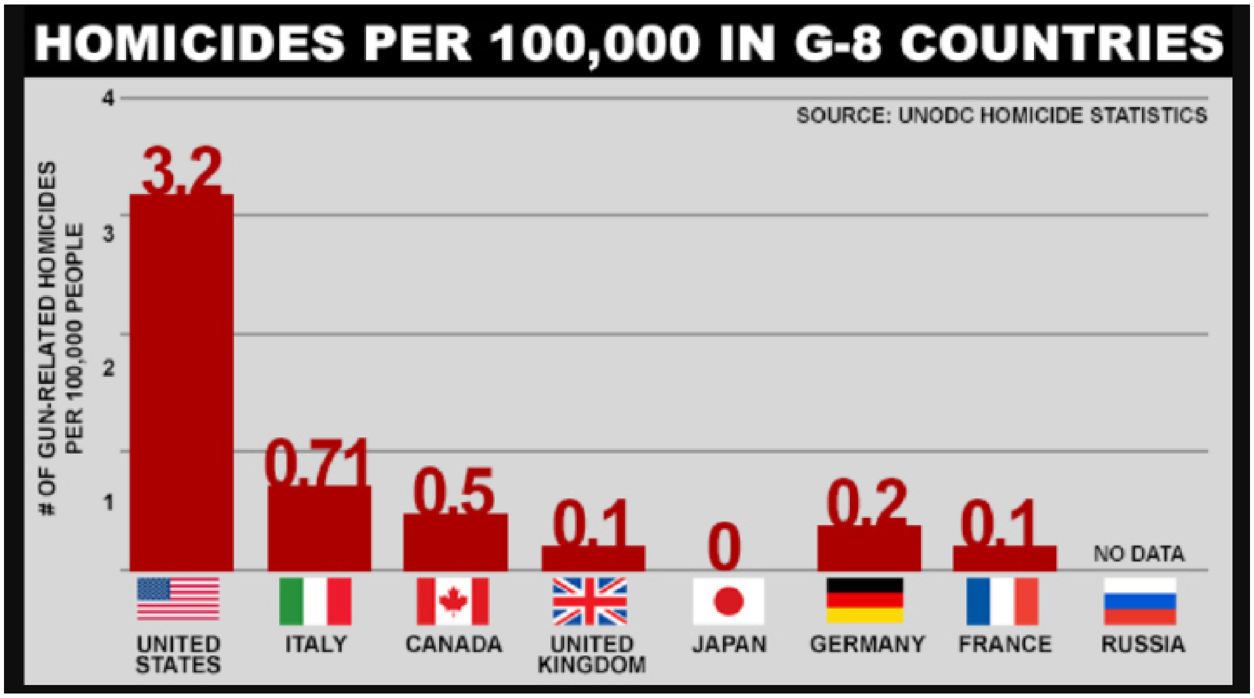


Figure 1: US gun homicides chart

Or even worse, this version from everytown.org:

But then you are reminded that the US is a large and diverse country with 50 very different states as well as the District of Columbia (DC).

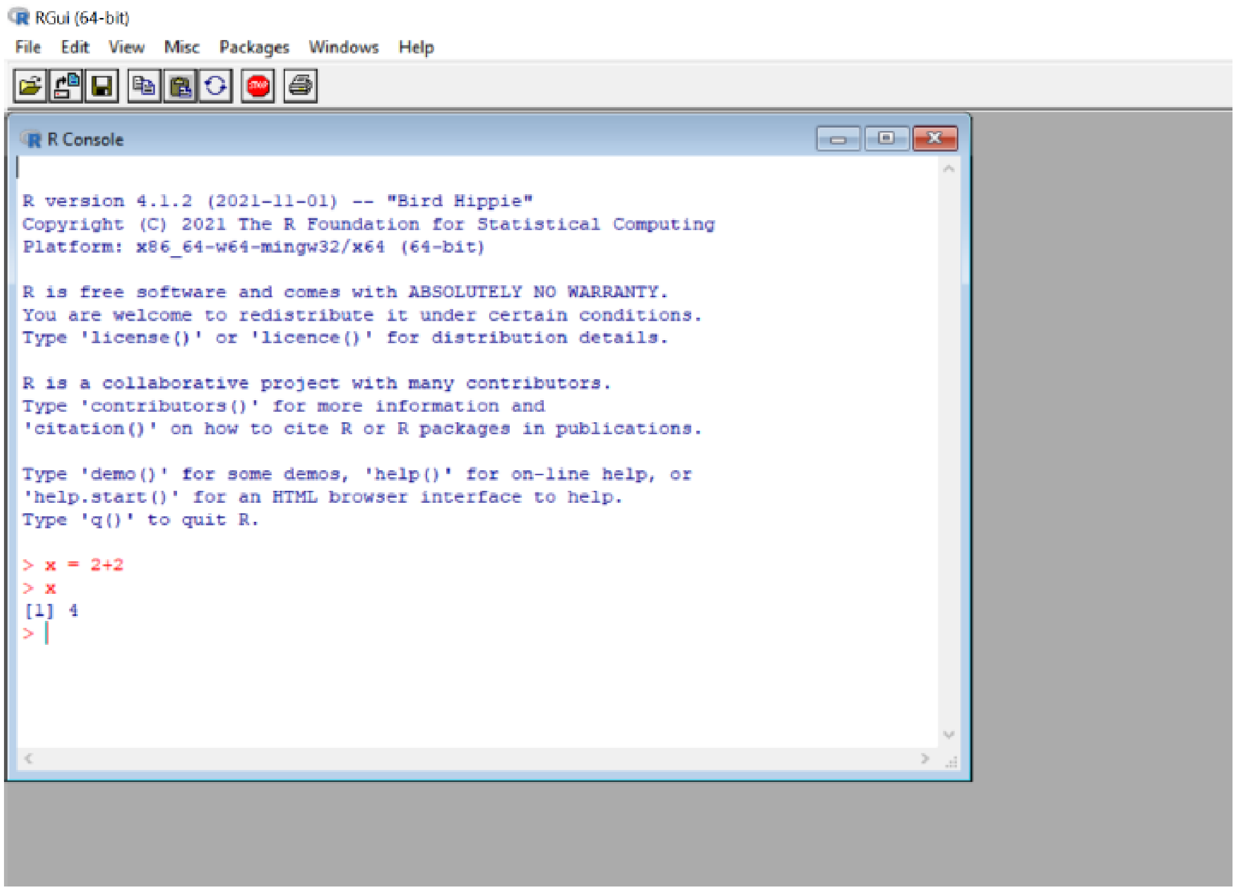
California, for example, has a larger population than Canada and 20 US states have populations larger than that of Norway. In some respects the variability across states in the US is akin to the variability across countries in Europe. Furthermore, although not in the charts above, the murder rates in Lithuania, Ukraine, and Russia are higher than 4 per 100,000. So perhaps the news reports that worried you are too superficial. You have options of where to live and want to find out how safe each state is. We will gain some insights by examining data related to gun homicides in the US using R.

Now before we get started with our example, we need to cover logistics as well as some of the very basic building blocks that we need to gain more advanced R skills. Be aware that for some of these, it is not immediately obvious how it is useful, but later in the book you will appreciate having the knowledge under your belt.

Introduction to R, R-Studio and R-Markdown

Introduction to R and R-Studio

R is a statistical programming language in which a user types commands to obtain desired results. In other words it is simply a big glorified calculator.



```
R version 4.1.2 (2021-11-01) -- "Bird Hippie"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

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.

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.

> x = 2+2
> x
[1] 4
>
```

Typing codes in RGUI is quite cumbersome in may aspects so developers came up with R studio which is simply an Interactive Development Environment (IDE) which is a fancier and eary way of writing R codes.

Why R?

- Lots and lots of Statistical libraries/packages.
- Awesome online community (stack overflow, kagggle, #rstats, #TidyTuesday etc): Probably >1000 people will have gotten the same error message by the time you get one.
- Reproducibility: You can share your codes and data and others are able to reproduce the results.

- Versatility: You can create your own functions
- It if free.

R-Studio

You do not have to use RStudio to access R, and many people don't!
Other IDEs that work with R include:

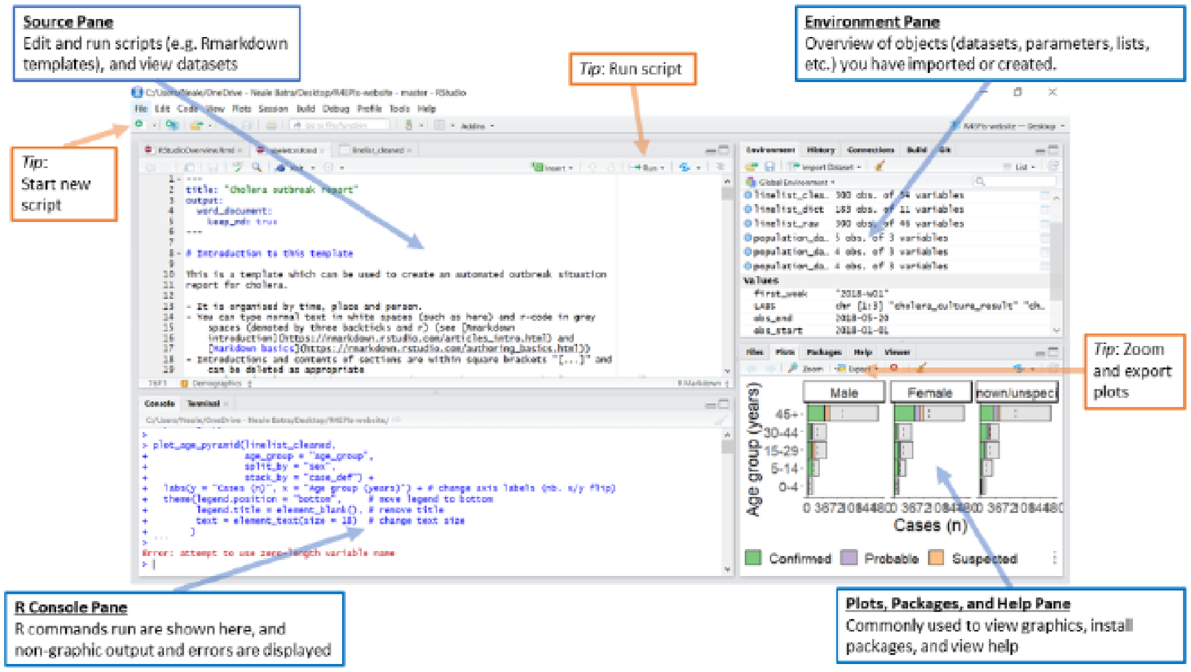
- Jupyter notebook (https://jupyter.org/)
- VisualStudio (https://visualstudio.microsoft.com/services/visual-studio-online/)
- VIM (https://github.com/jalvesaq/Nvim-R)

*IntelliJ IDEA (https://plugins.jetbrains.com/plugin/6632-r-language-for-intellij)

- EMACS Speaks Statistics (ESS) (https://ess.r-project.org/)

This is a non-exhaustive list, and most of these options require a good deal of familiarity with a given IDE.

RStudio Layout



Source Pane
Edit and run scripts (e.g. Rmarkdown templates), and view datasets

Tip: Run script

Environment Pane
Overview of objects (datasets, parameters, lists, etc.) you have imported or created.

Tip:
Start new script

Tip: Zoom and export plots

R Console Pane
R commands run are shown here, and non-graphic output and errors are displayed

Plots, Packages, and Help Pane
Commonly used to view graphics, install packages, and view help

```
---
title: "Cholera outbreak report"
output:
  word_document:
    keep_md: true
---

# Introduction to this template

This is a template which can be used to create an automated outbreak situation
report for cholera.

- It is organised by time, place and person.
- You can type normal text in white spaces (such as here) and r-code in grey
    spaces (denoted by three backticks and r) (see [Rmarkdown
    introduction](https://rmarkdown.rstudio.com/articles_intro.html) and
    [Markdown basics](https://rmarkdown.rstudio.com/authoring_basics.html))
- Introductions and contents of sections are within square brackets "[...]" and
    can be deleted as appropriate
```

```
>
> plot_age_pyramid(linelist_cleaned,
+                  age_group = "age_group",
+                  split_by = "sex",
+                  stack_by = "case_def") +
+   labs(y = "Cases (n)", x = "Age group (years)") + # change axis labels (nb. x/y flip)
+   theme(legend.position = "bottom",     # move legend to bottom
+         legend.title = element_blank(), # remove title
+         text = element_text(size = 18)  # change text size
+       )
+ ...
>
Error: attempt to use zero-length variable name
> |
```

```
linelist_clea…  300 obs. of 34 variables
linelist_dict   183 obs. of 11 variables
linelist_raw    300 obs. of 46 variables
population_da…  5 obs. of 3 variables
population_da…  4 obs. of 3 variables
population_da…  4 obs. of 3 variables
Values
  first_week    "2018-W01"
  LABS          chr [1:3] "cholera_culture_result" "ch…
  obs_end       2018-06-20
  obs_start     2018-01-01
```

If your RStudio displays only one left pane it is because you have no scripts open yet.

The Source Pane

The Source Pane This pane, by default in the upper-left, is a space to edit, run, and save your scripts. Scripts contain the commands you want to run.

The R Console Pane

The R Console, by default the left or lower-left pane in R Studio, is the home of the R "engine". This is where the commands are actually run and non-graphic outputs and error/warning messages appear. You can directly enter and run commands in the R Console, but realize that these commands are not saved as they are when running commands from a script.

The Environment Pane

This pane, by default in the upper-right, is most often used to see brief summaries of objects in the R Environment in the current session. These objects could include imported, modified, or created datasets, parameters you have defined, or vectors or lists you have defined during analysis (e.g. names of regions). You can click on the arrow next to a data frame name to see its variables.

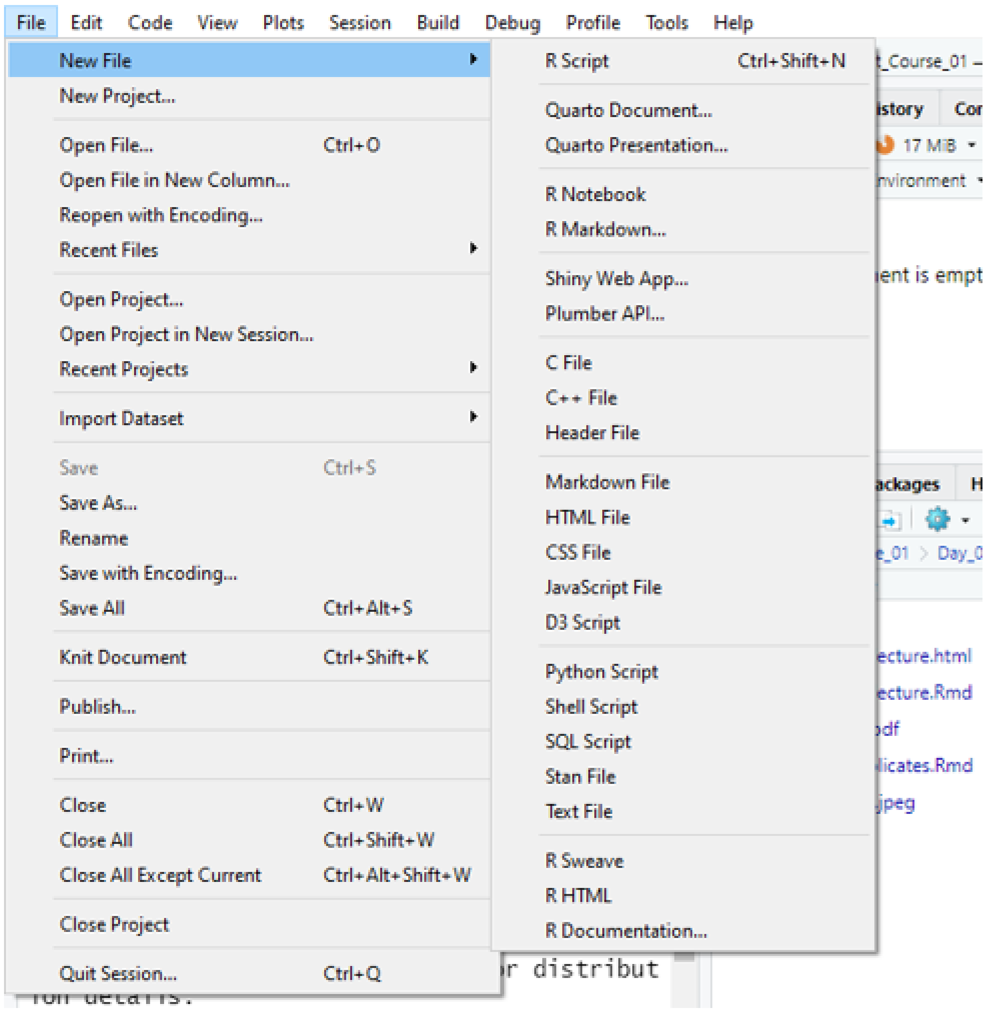
This pane also contains History where you can see commands that you can previously. It also has a "Tutorial" tab where you can complete interactive R tutorials if you have the learnr package installed. It also has a "Connections" pane for external connections, and can have a "Git" pane if you choose to interface with Github.

Plots, Viewer, Packages, and Help Pane

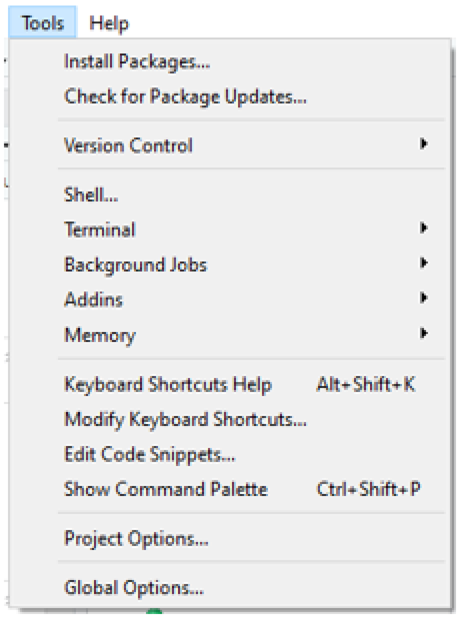
The lower-right pane includes several important tabs. Typical plot graphics including maps will display in the Plot pane. Interactive or HTML outputs will display in the Viewer pane. The Help pane can display documentation and help files. The Files pane is a browser which can be used to open or delete files. The Packages pane allows you to see, install, update, delete, load/unload R packages, and see which version of the package you have.

As you work with R more, you'll find yourself using the tabs within each of the panes.

We can open up an .R script in the source pane by going to "File", selecting "New File", and then selecting "R Script":



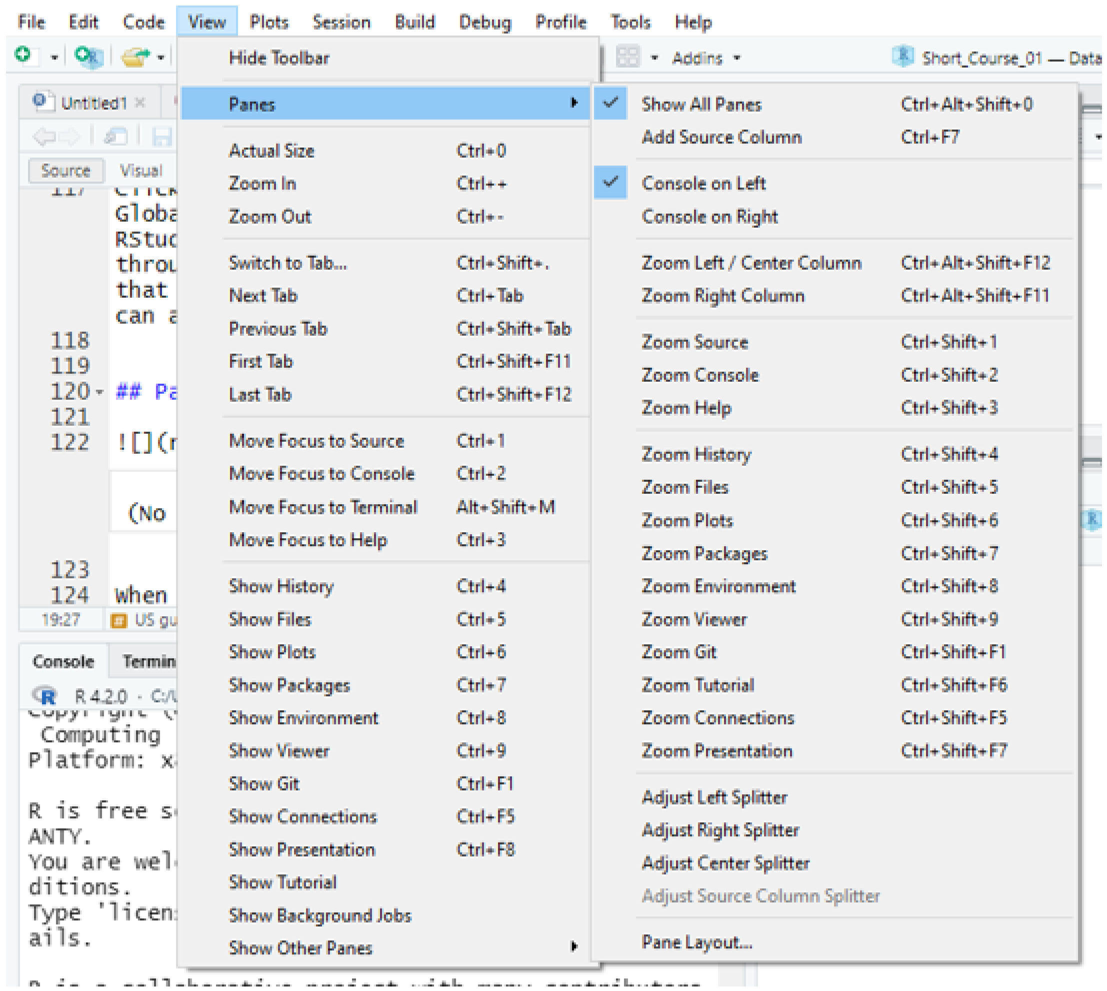
Customizing R Studio



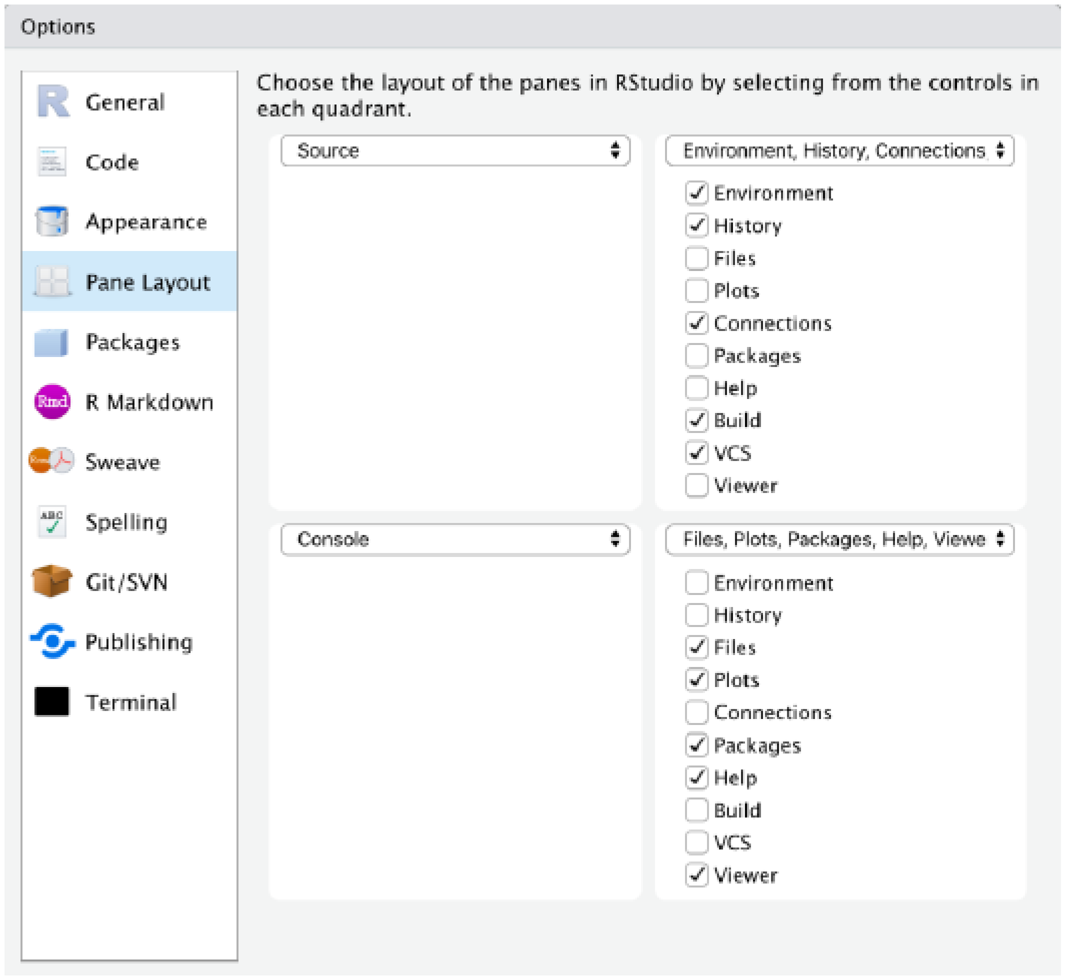
The "General" tab will open, with several checkboxes selected and unselected. From here, you can select your preferences.

Click on the "Appearance" tab from within Global Options. From here you can select your RStudio Font, Font Size, and Theme. Go through the options and select an appearance that works best for you, and know that you can always come back and change it!

Pane Layout



When we select Pane Layout, we'll see this:



Choose the layout of the panes in RStudio by selecting from the controls in each quadrant.

Source

Environment, History, Connections
- [✓] Environment
- [✓] History
- [ ] Files
- [ ] Plots
- [✓] Connections
- [ ] Packages
- [ ] Help
- [✓] Build
- [✓] VCS
- [ ] Viewer

Console

Files, Plots, Packages, Help, Viewe
- [ ] Environment
- [ ] History
- [✓] Files
- [✓] Plots
- [ ] Connections
- [✓] Packages
- [✓] Help
- [ ] Build
- [ ] VCS
- [✓] Viewer

From here, you can select which tabs you'd like to appear within each pane, and you can even change where each pane appears within RStudio. If our Environment Pane etc had been closed, we would select it from the Pane Layout in order to re-open it within RStudio.

Restart, Quit, New Session

If your R freezes, you can re-start R by going to the Session menu and clicking "Restart R". This avoids the hassle of closing and opening RStudio. Everything in your R environment will be removed when you do this.

You can open a new session and run new scrips if the current one is still running as some scripts can take days to execute.

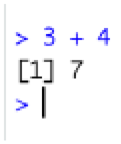
Writing and Running Codes in RStudio

Up to this point, we've been exploring the RStudio interface and setting up our preferences. Now, we'll shift to some basic coding practices. In order to run code in R, you need to type your code either in the Console or within an .R script.

We generally recommend creating an .R script as you're learning, as it allows you to type all of your code, add comments, and then save your .R script for reference. If instead you work entirely in the Console, anything that you type in the Console will disappear as soon as you restart or close R, and you will not be able to reference it in the future.

To run code in the Console, you type your code next to the > and hit Enter. We'll spend a little time practicing running code in the Console by exploring some basic properties of coding in R.

In the Console, type 3 + 4 and hit Enter. You should see the following:

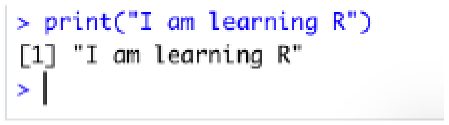


```
> 3 + 4
[1] 7
>
```

We've just used R to add the numbers 3 and 4. R has returned the sum of 3 + 4 on a new line, next to [1]. The [1] tells us that there is one row of data.

We can also use R to print out text. Type the following in the Console and hit Enter:

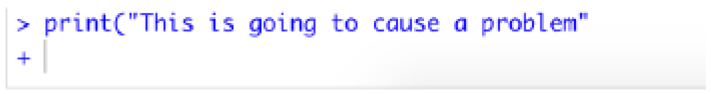
print("I am learning R")



```
> print("I am learning R")
[1] "I am learning R"
>
```

There's one error that you're likely going to come across, both when running code in the Console as well as in an R script. Let's explore that error now by running the following code in the Console and hitting Enter:

print("This is going to cause a problem"



```
> print("This is going to cause a problem"
+
```

When we're missing a closing parenthesis, R is expecting us to provide more code. We know this because instead of seeing a carat > in our Console, we see a +, and R has not returned the print statement that we were expecting! There are two ways to fix this problem:

- Type the closing) in the Console and hit Enter
- Hit the Esc key

Running Code in an R Script

The key to using the script editor effectively is to memorize one of the most important keyboard shortcuts: Cmd + Enter for Mac or Ctrl + Enter otherwise. This executes the current R expression in the console. For example, take the code below. If your cursor is at , pressing Cmd/Ctrl + Enter will run the complete command that generates fit. It will also move the cursor to the next statement (summary(fit)). That makes it easy to run your complete script by repeatedly pressing Cmd/Ctrl + Enter.

```
data("mtcars")

fit <- lm(mpg ~ disp + hp + drat,
          data = mtcars,
          weights = runif(nrow(mtcars)))

summary(fit)
```

```
##
## Call:
```

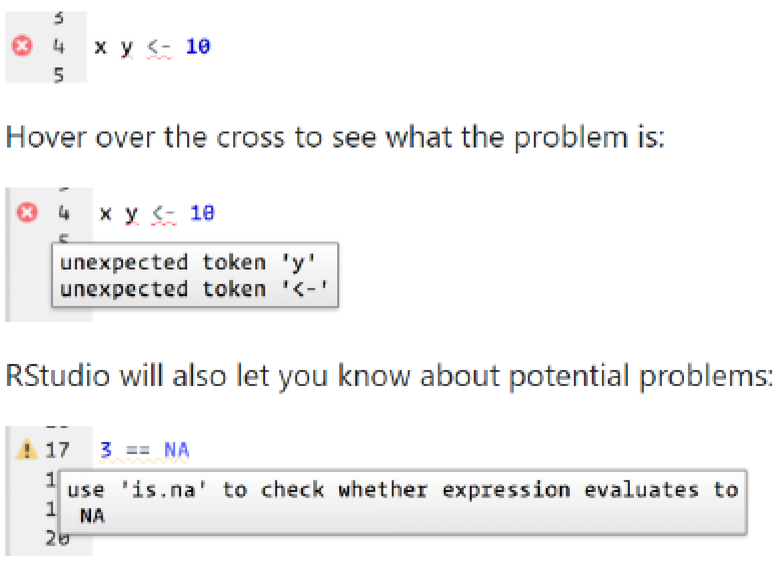
```
## lm(formula = mpg ~ disp + hp + drat, data = mtcars, weights = runif(nrow(mtcars)))
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7081 -1.1537 -0.1108  0.6223  5.9103
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 19.27795    6.96398   2.768  0.00988 **
## disp        -0.02471    0.01180  -2.095  0.04537 *
## hp          -0.02668    0.01612  -1.655  0.10911
## drat         2.87663    1.62467   1.771  0.08752 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.318 on 28 degrees of freedom
## Multiple R-squared:  0.7697, Adjusted R-squared:  0.7451
## F-statistic:  31.2 on 3 and 28 DF,  p-value: 4.523e-09
```

Instead of running expression-by-expression, you can also execute the complete script in one step: Cmd/Ctrl + Shift + S

It is always reccomended to start your script with packages that you need.

RStudio diagnostics

The script editor will also highlight syntax errors with a red squiggly line and a cross in the sidebar:



```
x y <- 10
```

Hover over the cross to see what the problem is:

```
x y <- 10
unexpected token 'y'
unexpected token '<-'
```

RStudio will also let you know about potential problems:

```
3 == NA
use 'is.na' to check whether expression evaluates to
 NA
```

Commenting Your R code.

It is considered good practice to comment your code when working in an .R script. Even if you are the only person to ever work on your code, it can be helpful to write yourself notes about what you were trying to do with a specific piece of code. Moreover, writing comments in your code as you work through the examples in this book is a great way to help reinforce what you're learning. Comments are ignored by R when running a script, so they will not affect your code or analysis.

To comment out a line of code, you can place a pound sign (also called an octothorpe) # in front of the line of code that you want to exclude when you're running your script.

If you think you'll be writing more than one line of comments, you can do a pound sign followed by a single quotation mark (#'). This will continue to comment out lines of text or code each time you hit Enter. You can delete the #' on a new line where you want to write code for R to run. This method is useful when you're writing a long description of what you're doing in R.

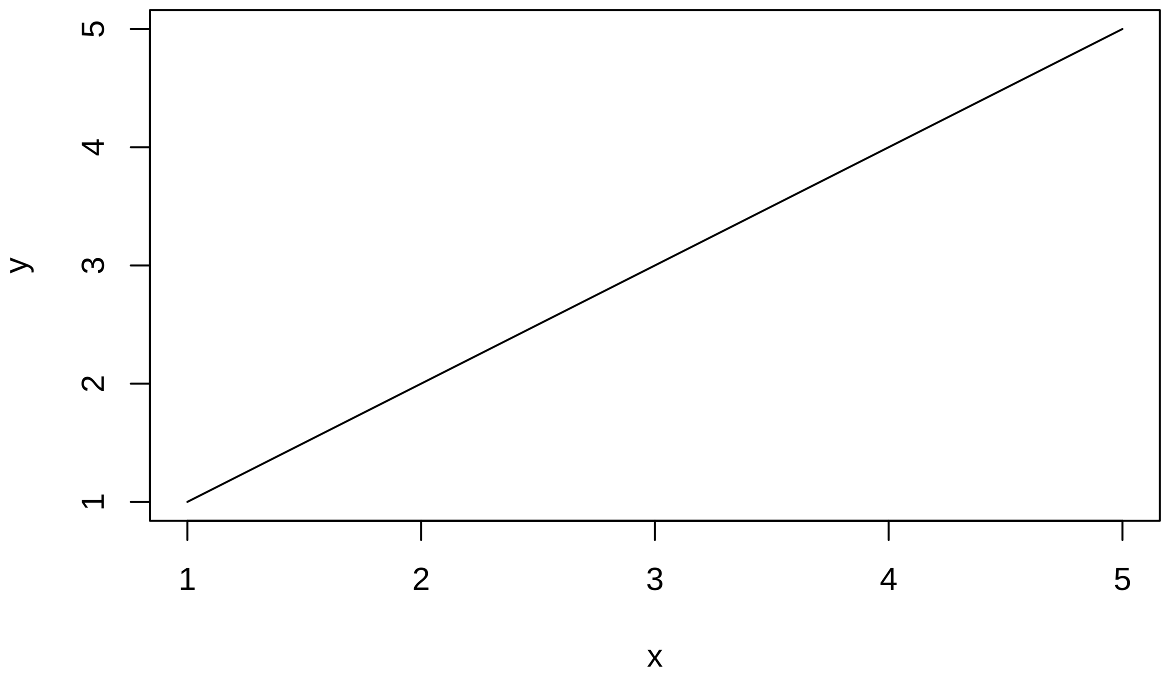
#' this will be a short code example. #' you are not expected to know what this does, #' nor do you need to try running it on your computer. library(readr) # load the readr package library(here) # load the here package data <- read_csv(here("file_path", "file_name.csv")) # save file_name.csv as data

Autocomplete to avoid coding mistakes

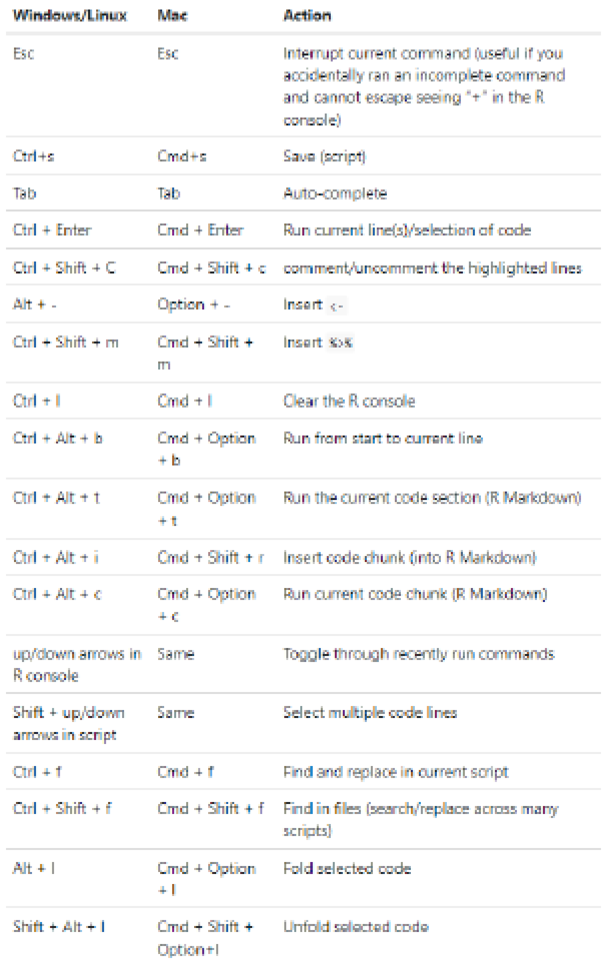
Use your Tab key when typing to engage RStudio's auto-complete functionality. This can prevent spelling errors, remind you of function arguments.

Press Tab while typing to produce a drop-down menu of likely functions and objects, based on what you have typed so far.

```
#install.packages("splines2")
library(splines)
plot(x = 1:5,y = 1:5, type = "l",xlab = "x",ylab = "y")
```



Keyboard Shortcuts



Windows/Linux	Mac	Action
Esc	Esc	Interrupt current command (useful if you accidentally ran an incomplete command and cannot escape seeing "+" in the R console)
Ctrl+s	Cmd+s	Save (script)
Tab	Tab	Auto-complete
Ctrl + Enter	Cmd + Enter	Run current line(s)/selection of code
Ctrl + Shift + C	Cmd + Shift + c	comment/uncomment the highlighted lines
Alt + -	Option + -	Insert `<-`
Ctrl + Shift + m	Cmd + Shift + m	Insert `%>%`
Ctrl + l	Cmd + l	Clear the R console
Ctrl + Alt + b	Cmd + Option + b	Run from start to current line
Ctrl + Alt + t	Cmd + Option + t	Run the current code section (R Markdown)
Ctrl + Alt + i	Cmd + Shift + r	Insert code chunk (into R Markdown)
Ctrl + Alt + c	Cmd + Option + c	Run current code chunk (R Markdown)
up/down arrows in R console	Same	Toggle through recently run commands
Shift + up/down arrows in script	Same	Select multiple code lines
Ctrl + f	Cmd + f	Find and replace in current script
Ctrl + Shift + f	Cmd + Shift + f	Find in files (search/replace across many scripts)
Alt + l	Cmd + Option + l	Fold selected code
Shift + Alt + l	Cmd + Shift + Option+l	Unfold selected code

Resources for learning R and Rstudio

Help Documentation

Functions, datasets, and other built-in objects in R are documented in its help system. Search the RStudio "Help" tab for documentation on R packages and specific functions. This is within the pane that also contains Files, Plots, and Packages (typically in the lower-right pane). As a shortcut, you can also type the name of a package or function into the R console after a question-mark to open the relevant Help page. Do not include parentheses.

For example: ?mean or ?plot or help(plot).

The quality of R's help pages varies somewhat with most tending to be very brief. The figure below provides an overview of what to look for. Remember, functions take inputs, perform actions, and return outputs. Something goes in, it gets worked on, and then something comes out. That means you want to know what the function requires, what it does, and what it returns. What it requires is shown in the Usage and Arguments sections of the help page. The names of all the required and optional arguments are given by name and in the order the function expects them. Some arguments have default values. In the case of the mean() function the argument na.rm is set to FALSE by default. These will be shown in the Usage section. If a named argument has no default, you will have to give it a value. Depending on what the argument is, this might be a logical value, a number, a dataset, or any other object.

Online Forums

- StackOverflow: (a Q&A site with hundreds of thousands of answers to all sorts of programming questions)
- RStudio Community (a forum specifically designed for people using RStudio and the tidyverse.
- Searching for help with R on Google can sometimes be tricky because the program name is a single letter. Google is generally smart enough to figure out what you mean when you search for "r scatterplot", but if it does struggle, try searching for "rstats" instead (e.g. "rstats scatterplot").

Cheatsheets

- There are many PDF "cheatsheets" available on the RStudio website!

Other online souces

- Twitter
- Youtube