

# Module 3: R

## Hello, World

Instructor: Anjali Silva, PhD

TA: Tia Harrison, MSc

Data Sciences Institute, University of Toronto

27 June 2022

# Course Documents

- Visit: <https://github.com/anjalisilva/IntroductionToR>
- All course material will be available via IntroductionToR GitHub repository (<https://github.com/anjalisilva/IntroductionToR>). Folder structure is as follows:
  - Lessons - All files: This folder contains all files.
  - **Lessons - Data only**: This folder contains data only.
  - **Lessons - Lesson Plans only**: This folder contains lesson plans only.
  - **Lessons - PDF only**: This folder contains slide PDFs only.
  - README - README file
  - .gitignore - Files to ignore specified by instructor

# Course Contacts

- Instructor: Anjali Silva Email: [a.silva@utoronto.ca](mailto:a.silva@utoronto.ca) (Must use the subject line DSI-IntroR. E.g., DSI-IntroR: Inquiry about Lecture I.)
- TA: Tia Harrison Email: [tia.harrison@mail.utoronto.ca](mailto:tia.harrison@mail.utoronto.ca)

# Overview

Getting set-up (Alexander (eds), 2021, Chapters 2-4)

- R
- RStudio

R basics (Wickham and Grolemund, 2017, Chapter 4)

File types

- scripts (Wickham and Grolemund, 2017 Chapter 6)
- RMarkdown (Wickham and Grolemund, 2017 Chapter 27)

Hello, World!

# Getting set-up

# R

What is R?

- A programming language focused on statistics
- Open source and free

## Getting R

1. Go to <https://cloud.r-project.org/>
2. Click *Download R for (your operating system)*.
3. For Windows, select *install R for the first time*. For Mac and Linux, select the download that is appropriate for your OS.
4. Download and install.

# RStudio

What is RStudio?

- The integrated development environment (IDE) that lets us run R code
- Desktop and cloud versions available

## Getting RStudio

1. Go to [rstudio.com](https://rstudio.com)
2. Under *Products*, look under *Open Source* and select \*RStudio.
3. Scroll down and select *RStudio Desktop* and *DOWNLOAD RSTUDIO DESKTOP*.
4. Select the *DOWNLOAD* button under the *Free* version of RStudio Desktop.
5. If the download that is "Recommended for your system" is correct, click the download button. If not, scroll down and find the version that is correct for your OS.
6. Download and install.
7. Open and test to ensure RStudio is working.

**Any questions with R or  
RStudio setup?**

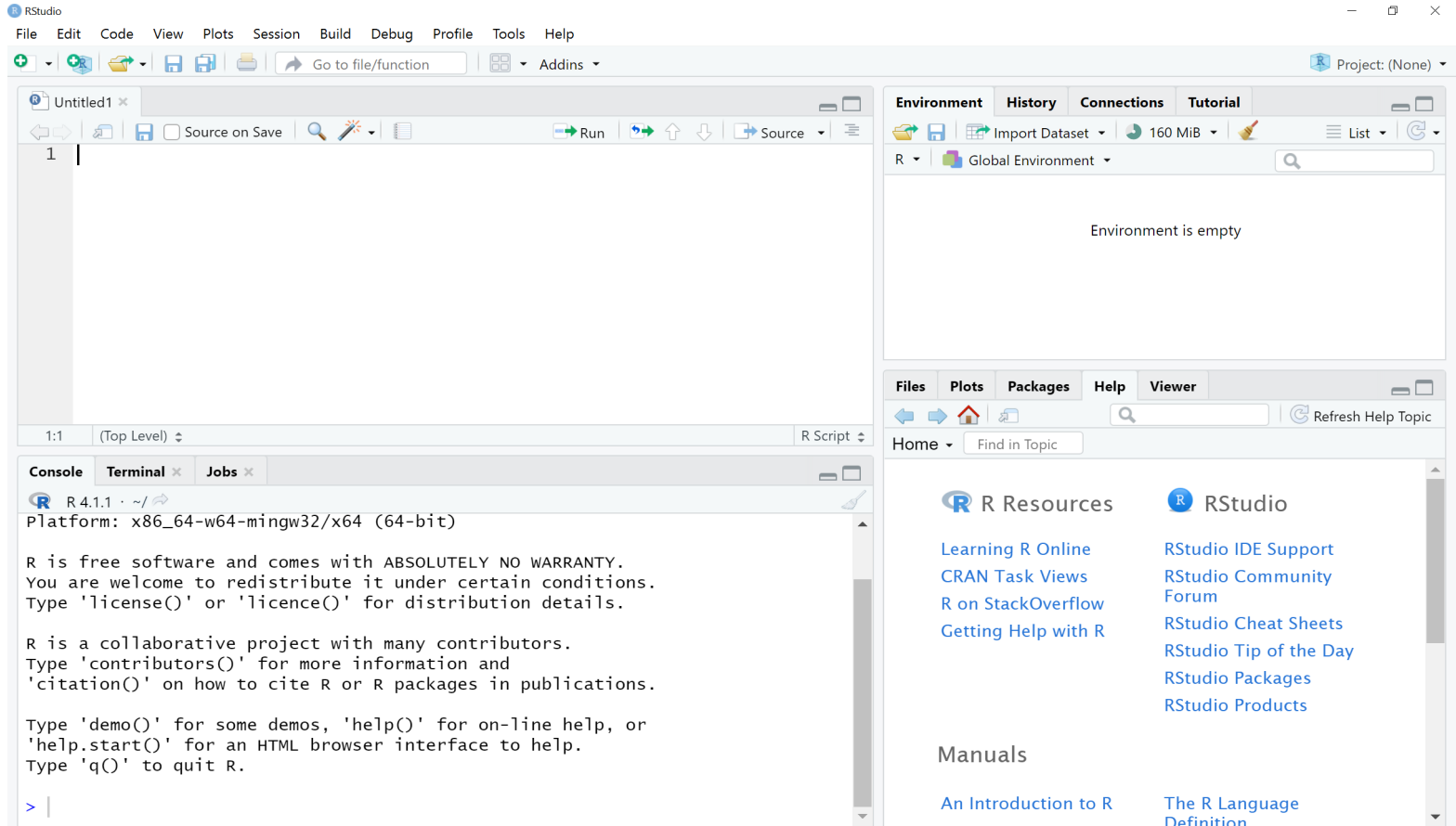
# If you have downloaded R and RStudio:

- Read about RStudio: <http://swcarpentry.github.io/r-novice-inflammation/09-supp-intro-rstudio/index.html>
- Watch the video about data science tools. Pay attention to R: <https://www.youtube.com/watch?v=pKPaHH7hmv8&t=99s>

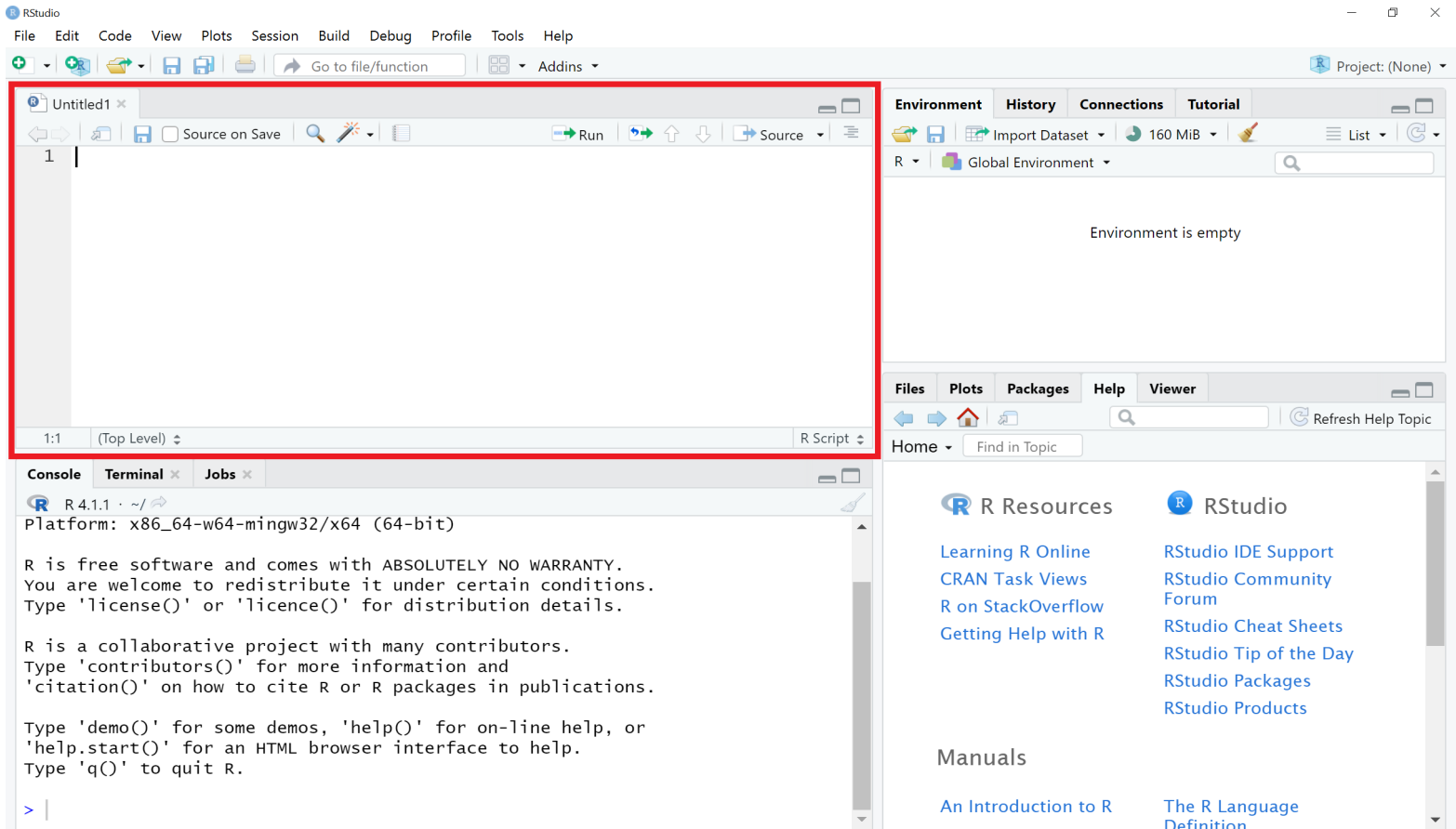


**RStudio**

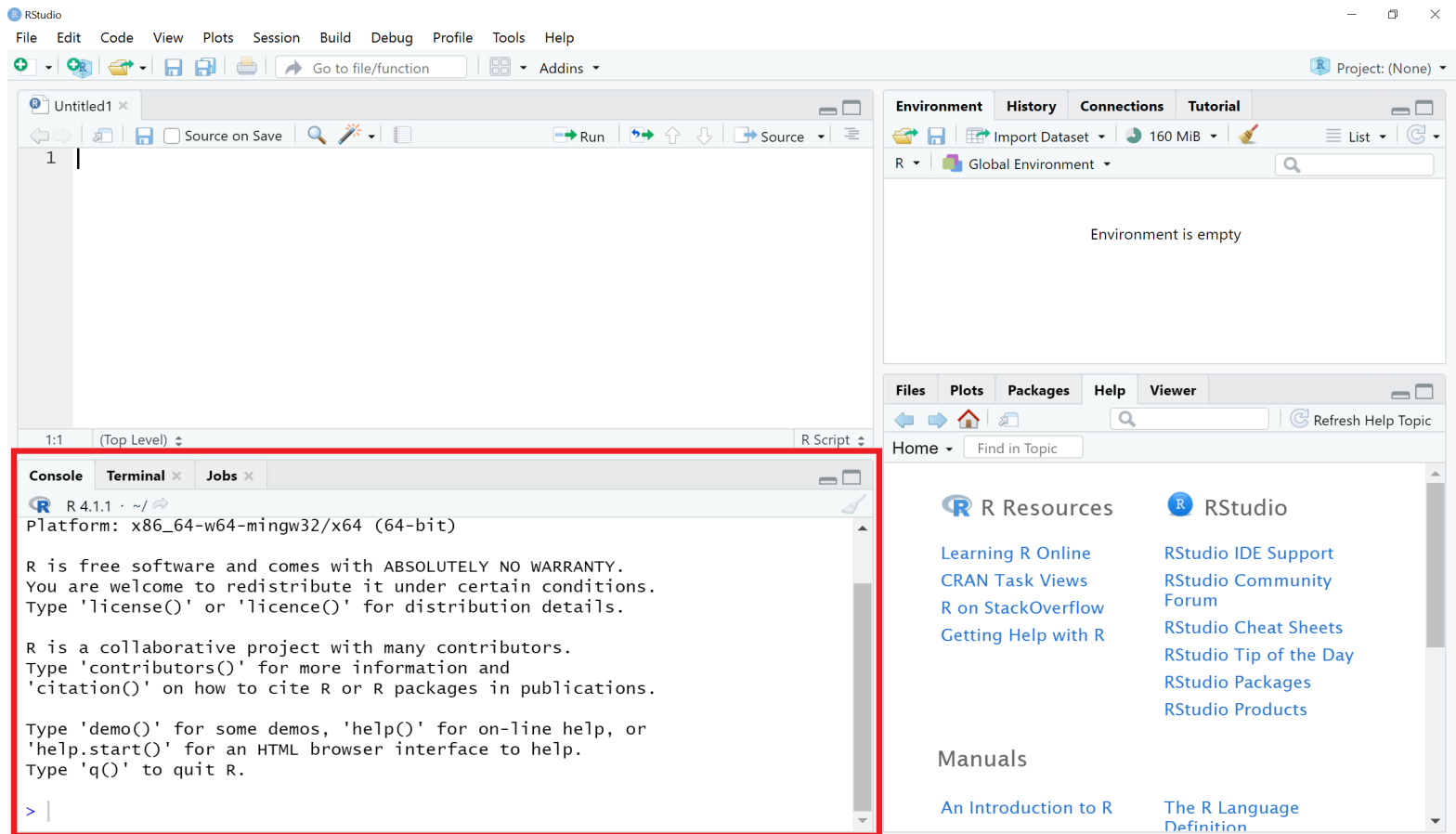
# Welcome to RStudio



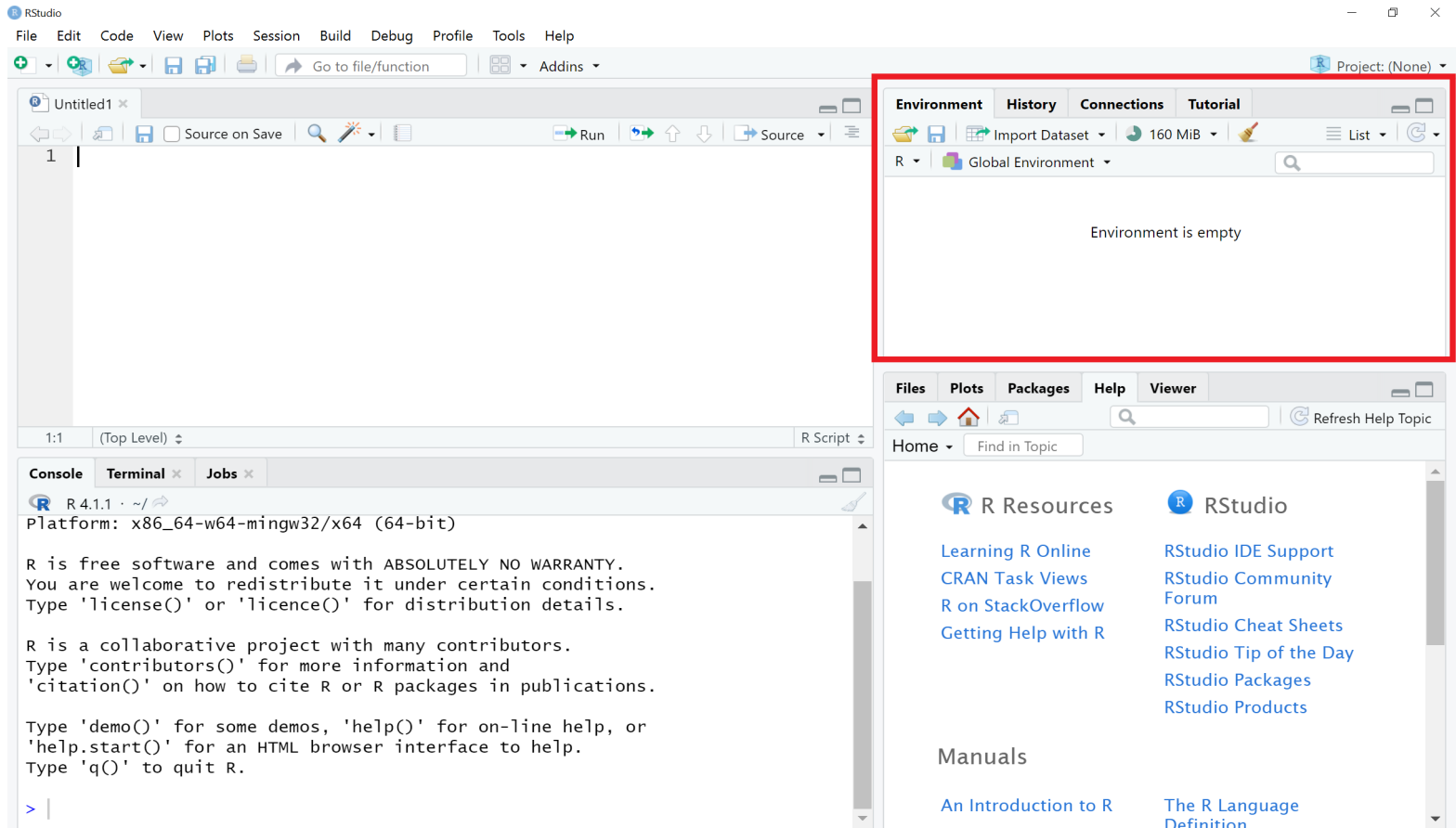
Your saved R files, where you can write and edit code, are in the upper left. If you do not have a file open, this section will be collapsed.



The console, where you run code and view output, is on the lower left.

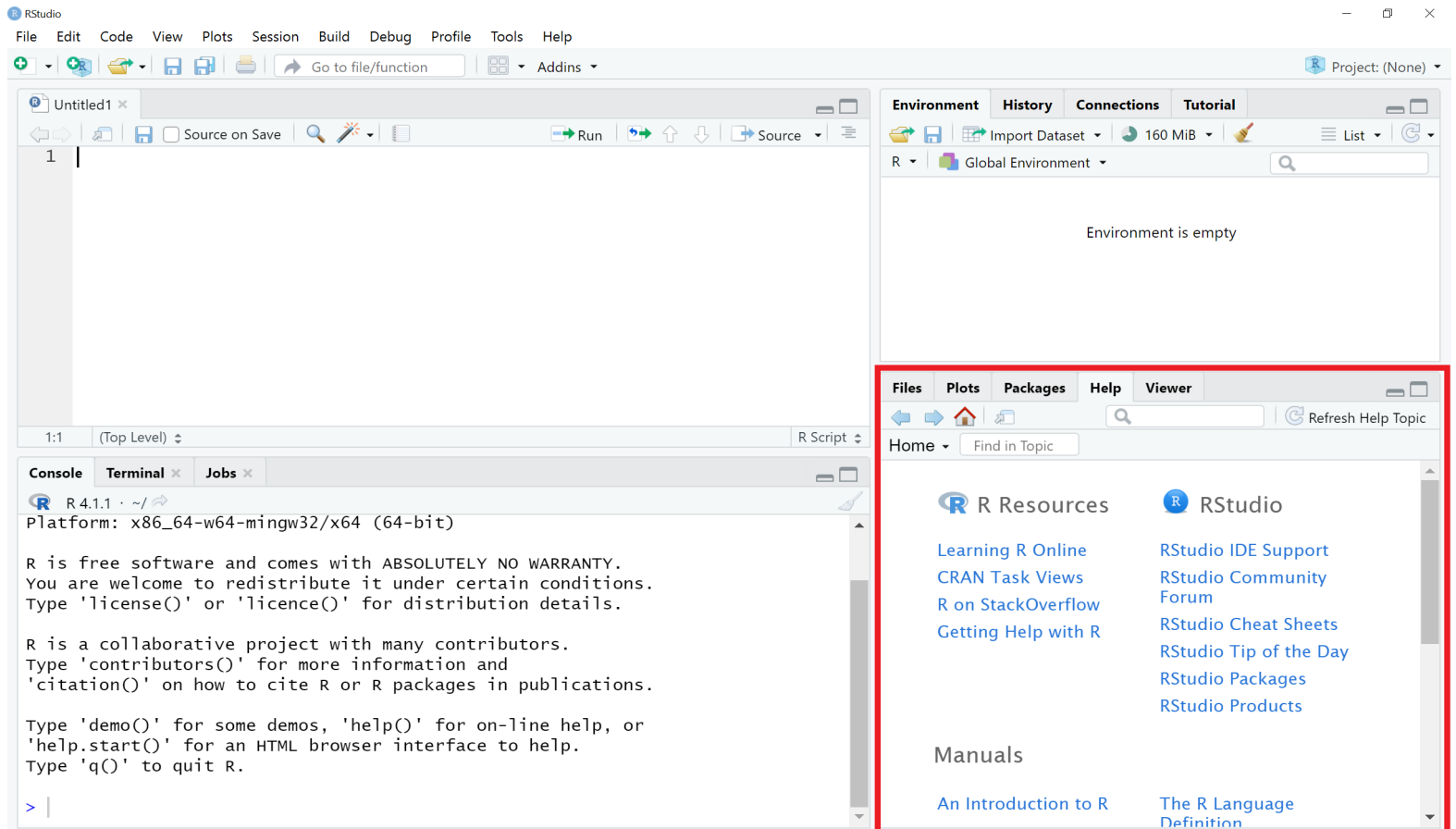


The environment, where you can see variables that you have saved, is on the upper right.



The lower right section contains:

- *Files*, where you see the folder structure of your working directory
- *Plots*, where plots you create are displayed
- *Packages*, an inventory of all R packages that are installed
- *Help*, where you can search information about R functions and packages



# Interacting with R

- Console:
  - Type commands directly into the console and press 'Enter' to execute.
- Script:
  - Put cursor at the end of the line to execute OR highlight the section.
  - Press 'Ctrl' + 'Enter' on Windows, Mac OR 'Cmd' + 'Return' on Mac.
- Clear console with 'Ctrl' + 'L'.
- If R is still waiting for you to enter more text, the console will show a + prompt.

# R Project

- Good to keep data, analyses, and text in a single folder.
- RStudio interface for this is Projects.
  - File → New project; choose New directory → New project
- Enter a name for this new folder (“directory”) and choose a convenient location for it. This will be your working directory.
  - On Desktop, save as ‘DSI IntroR’
- Click on ‘Create’ project.
- Create a new file where we will type our scripts.
  - Go to File → New File → R script. Click the save icon on your toolbar and save your script as “script.R”.



# R basics

# Location

- Current location:

```
getwd() # current location of the file, if saved
```

- Set working directory: By typing the path

```
setwd("/Users/<Name>/Desktop")
```

- Or (recommended method):
  - Session → Set Working Directory → Choose Directory...

# R Coding Style

- Limit yourself to 80 characters per line.
- Use comments. Don't describe what the code does, but explain why you wrote it that way.
- Use only `←` for assignment, not `=`.
- Never reassign reserved words.
- You may read more:
  - <https://google.github.io/styleguide/Rguide.html>
  - [http://steipe.biochemistry.utoronto.ca/abc/index.php/RPR-Coding\\_style](http://steipe.biochemistry.utoronto.ca/abc/index.php/RPR-Coding_style)

# R Features

- In R, the indexing begins from 1.
- R is case sensitive (“X” is not the same as “x”).
- R uses dynamic variable typing, so variables can be used over and over again.

# Mathematical operators

For now we will work in the console.

To run code, hit enter.

If it runs successfully, you will see a `>` on the line with the cursor.

If it instead shows a `+`, the command was incomplete. You can finish the command and hit enter, or hit ESCAPE to start again.

```
(27 + 0.4 - 2 * 7 / 11) ^ 3
```

```
## [1] 17835.37
```

# Creating objects

`<-` is the assignment operator.

To assign the value 27 to the object named `num_participants`, you type:

```
num_participants <- 27
```

Remember that names:

- must start with a letter
- can only contain letters, numbers, underscores, and periods

Running the name of the object will display the object.

```
num_participants
```

```
## [1] 27
```

# Functions

Built-in functions perform many operations. They take the form:

`function_name(argument1 = value1, argument2 = value2, ...)`

```
sqrt(16)
```

```
## [1] 4
```

```
seq(1, 14)
```

```
## [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14
```

# R Help

- Online documentation for functions and variables in R exists.
- Obtained by typing `help(FunctionName)` or `?FunctionName` at the R prompt, where `FunctionName` is name of function.

```
help(sqrt)
```

```
?sqrt
```



# R Packages

- Packages are collections of R functions, data, and compiled code.
- Libraries are directories in R where the packages are stored.
- Built-in functions are part of R standard or base packages and do not need to be downloaded.

```
library(help = "base")  
library(help = "stats")
```

- Some functions are not built-in. To get these, need to download packages
- R packages extend R's functionality.

# R Packages

- Popular repositories for Packages:
  - The Comprehensive R Archive Network (CRAN)
    - Link: <https://cran.r-project.org/web/packages/>
  - Bioconductor
    - Link: <https://www.bioconductor.org/packages/release/bioc/>
  - GitHub
    - Link: <https://github.com/search?q=r+packages&type=registrypackages>
- Depending on the source of Package, downloading instructions may differ.

# R Packages

- R packages extend R's functionality.
- tidyverse is a package from CRAN.
- Link: <https://cran.r-project.org/web/packages/tidyverse/index.html>
  - What is the current version?
  - Who is the author?
  - How to report an issue?
  - How can a user access Reference Manual?
- To download tidyverse:

```
install.packages("tidyverse")
```

- Every time you download a package or start a new RStudio session, you will need to load the packages you want to use.

```
library(tidyverse)
```

# tidyverse



## R packages for data science

The tidyverse is an opinionated **collection of R packages** designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

Install the complete tidyverse with:

```
install.packages("tidyverse")
```

Figure: Tidyverse R Packages designed for data science. Figure from <https://www.tidyverse.org/>.

- More details on 'tidyverse' package (only after downloading):

```
ls("package:tidyverse") # list all functions in package  
?tidyverse # get information on package
```

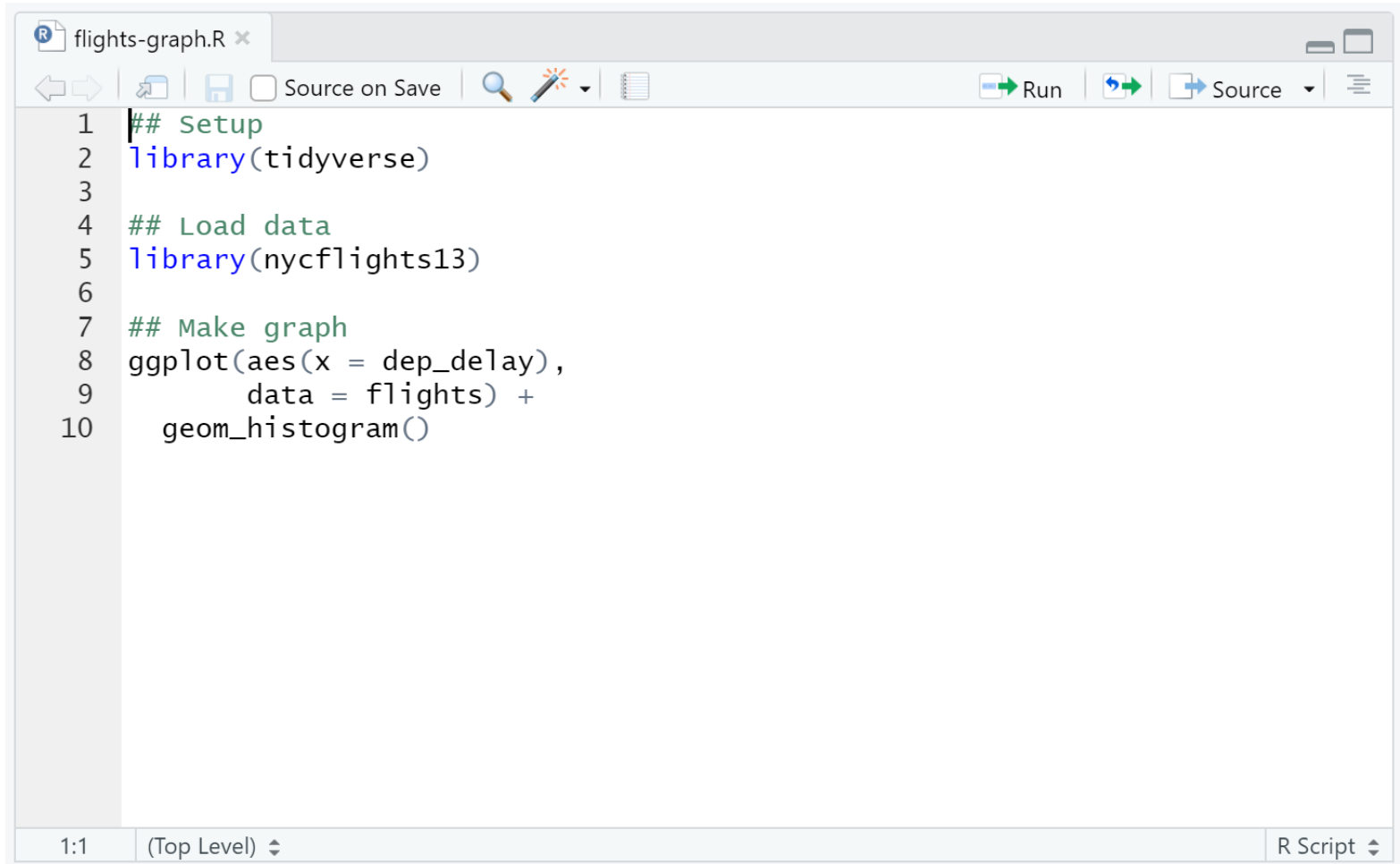
# Exercises

# Basic Operations

1. Install and load the `faraway` library.
2. Create an object named `my_sequence` that is a sequence from 1 to 7.
3. Use an R function to take the square roots of all the numbers in the sequence. Save this new sequence as an object named `sqrt_sequence`.
4. Multiply `sqrt_sequence` by 3.

# File types

# R Scripts



```
1 ## Setup
2 library(tidyverse)
3
4 ## Load data
5 library(nycflights13)
6
7 ## Make graph
8 ggplot(aes(x = dep_delay),
9         data = flights) +
10   geom_histogram()
```

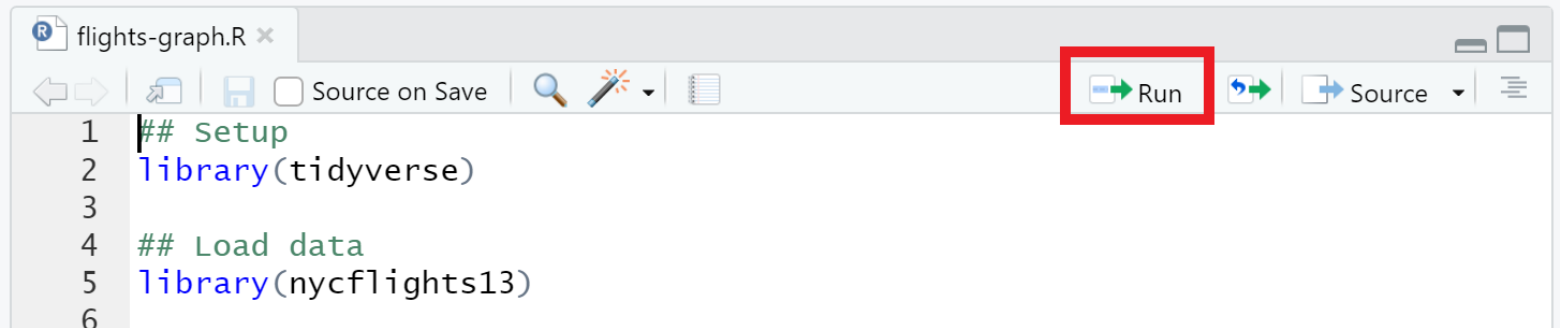


# R Scripts

R Scripts are files where you can save and edit your code.

## Running code

Run the entire script by pressing Run



Run the command where your cursor is located by pressing Cmd/Ctrl + Enter

Run a section of commands by highlighting them and pressing Cmd/Ctrl + Enter

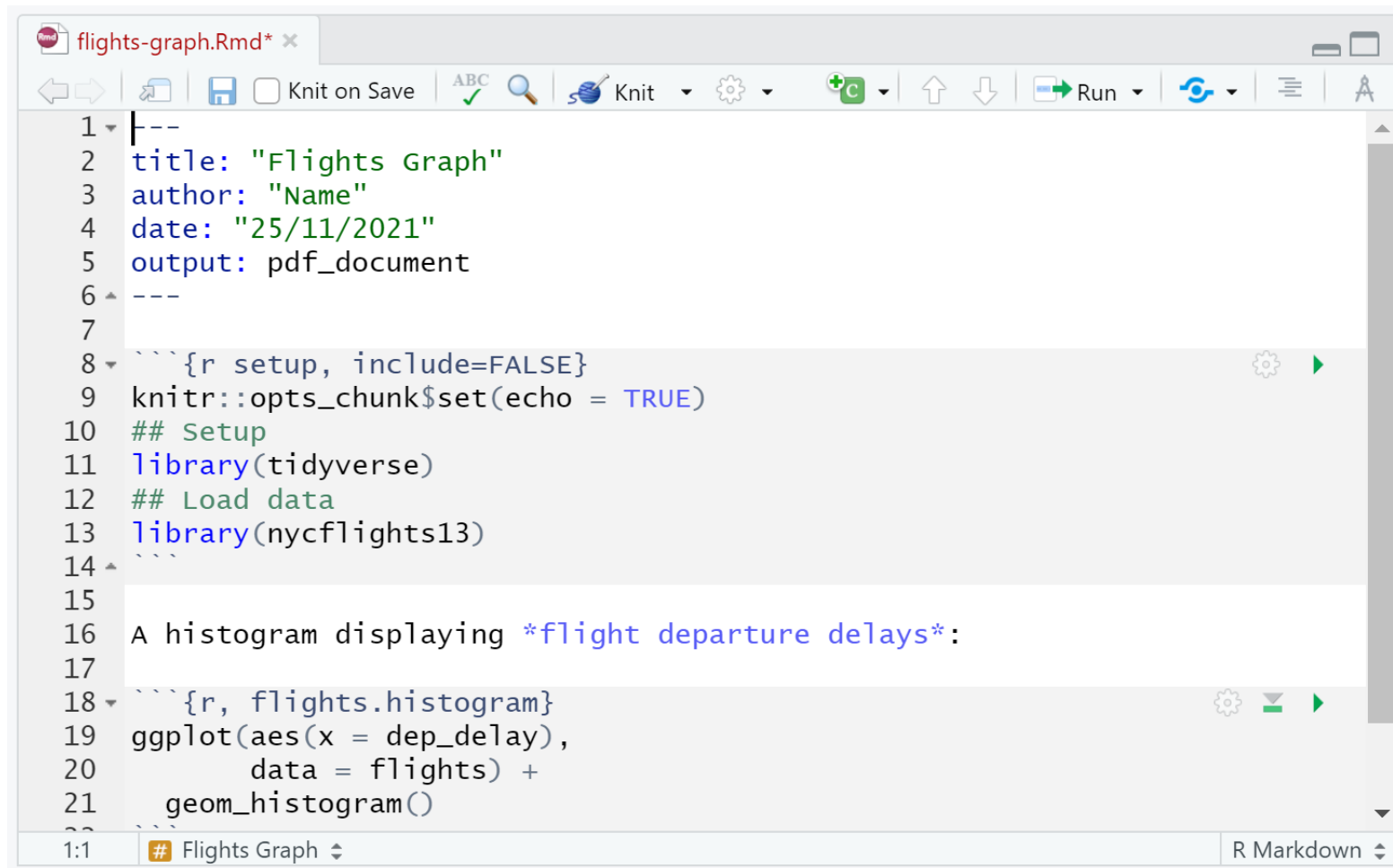
# Diagnostics

When working in a script, RStudio will mark syntax errors. If you hover over the red x, you can see what the problem is.

```
7  ## Make graph
8  ggplot(aes(x = dep_delay),
9          data = flights) +
10  geom_histogram(
```

unexpected end of document  
unmatched opening bracket '('

# R Markdown



The screenshot shows an R Markdown editor window with a single document titled "flights-graph.Rmd". The editor has a toolbar at the top with icons for navigation, saving, searching, knitting, and running. The code is as follows:

```
1 |<--
2 title: "Flights Graph"
3 author: "Name"
4 date: "25/11/2021"
5 output: pdf_document
6 ---
7
8 ```{r setup, include=FALSE}
9 knitr::opts_chunk$set(echo = TRUE)
10 ## Setup
11 library(tidyverse)
12 ## Load data
13 library(nycflights13)
14 ```
15
16 A histogram displaying flight departure delays:
17
18 ```{r, flights.histogram}
19 ggplot(aes(x = dep_delay),
20         data = flights) +
21   geom_histogram()
22 ```
23
24 1:1 # Flights Graph
```

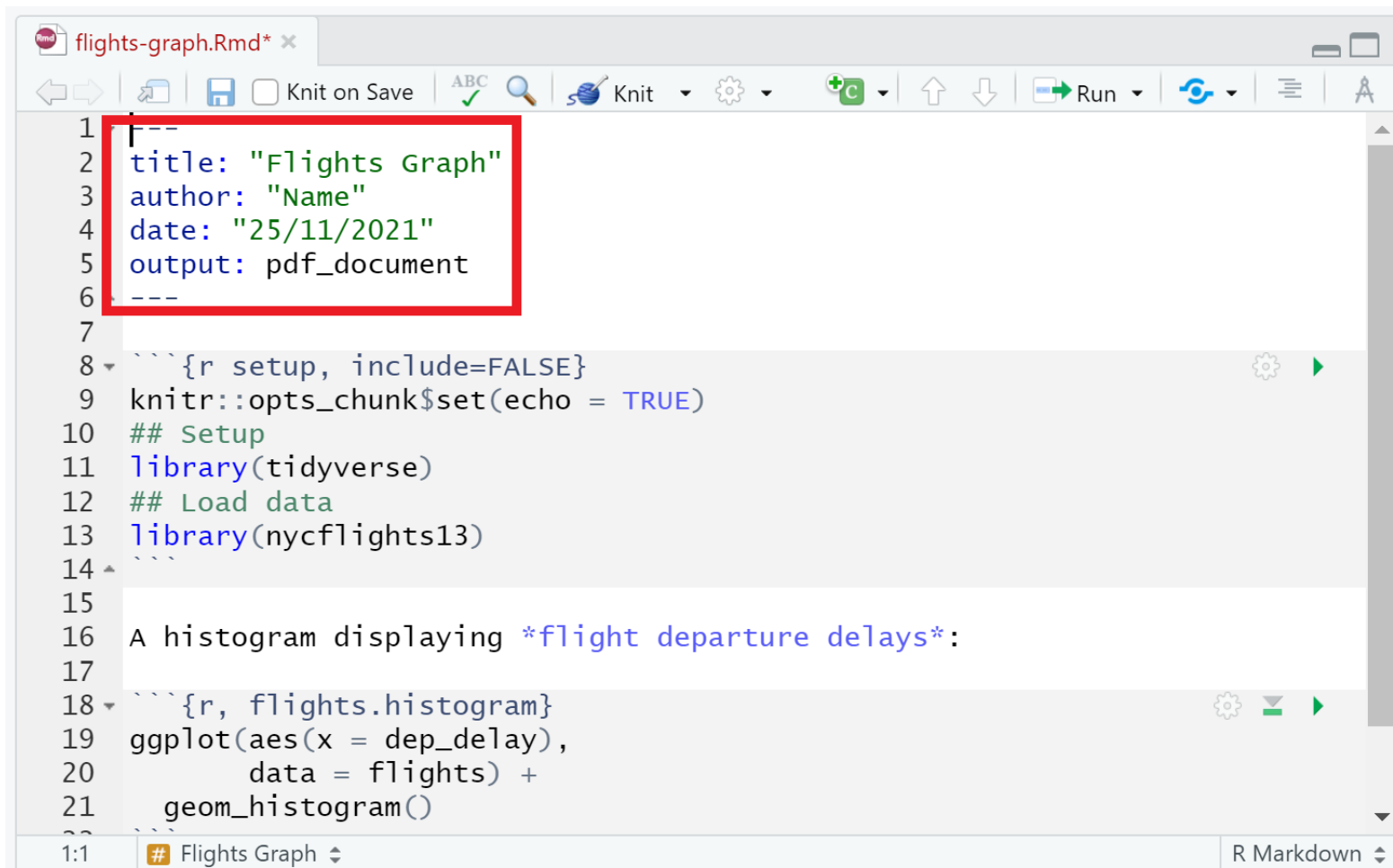
The status bar at the bottom indicates the current line is 1:1 and the document is an R Markdown file.

# R Markdown

R Markdown files combine code chunks with the results of those chunks and text. They support output formats like PDFs, html, Word files, and slideshows.

# Components

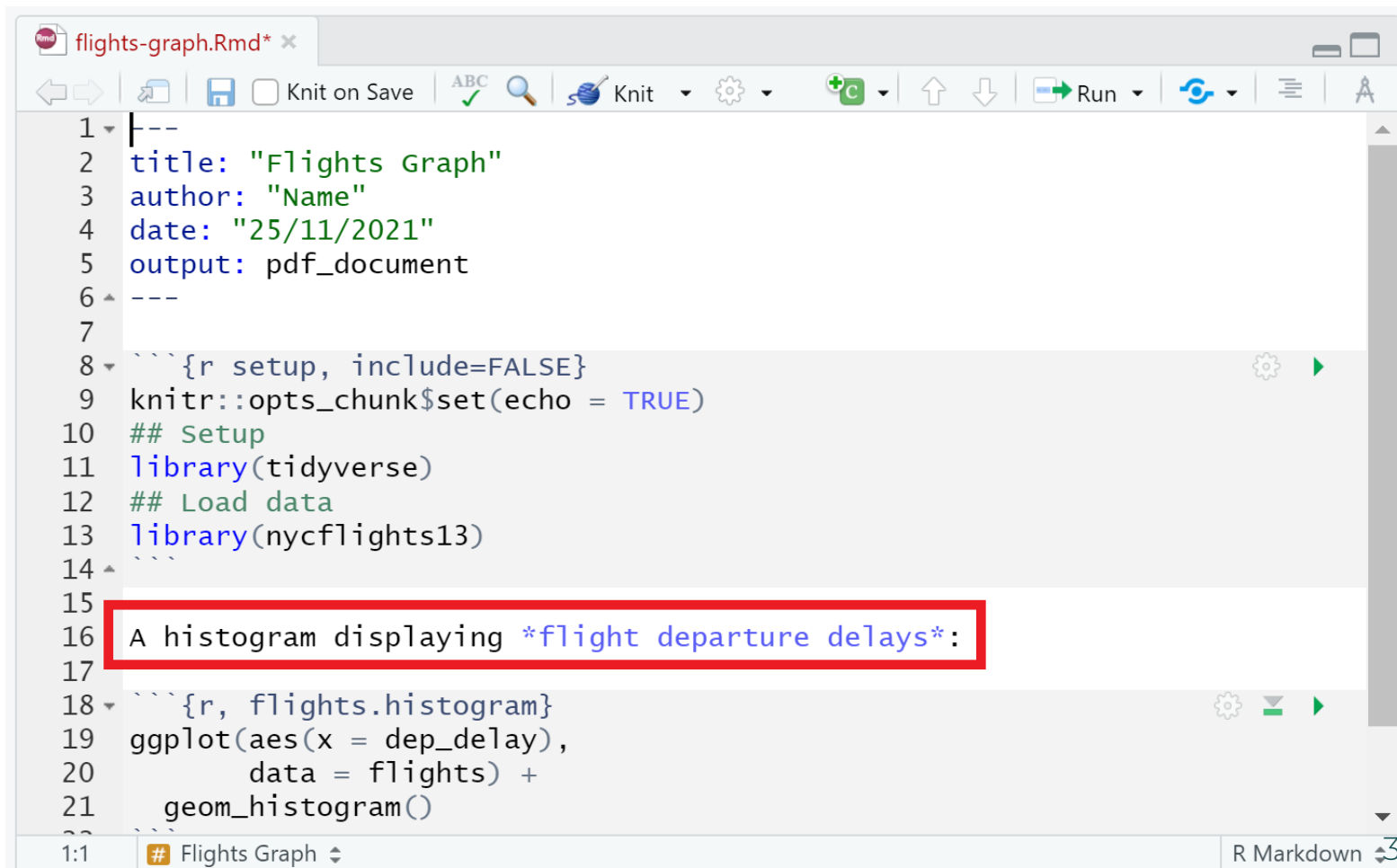
In the YAML header, the document information and settings are specified.



```
1 ---
2 title: "Flights Graph"
3 author: "Name"
4 date: "25/11/2021"
5 output: pdf_document
6 ---
7
8 {r setup, include=FALSE}
9 knitr::opts_chunk$set(echo = TRUE)
10 ## Setup
11 library(tidyverse)
12 ## Load data
13 library(nycflights13)
14
15
16 A histogram displaying flight departure delays:
17
18 {r, flights.histogram}
19 ggplot(aes(x = dep_delay),
20         data = flights) +
21   geom_histogram()
22
23 # Flights Graph
```

# Components

Text goes in between the code chunks. This text can be formatted with basic markdown syntax.

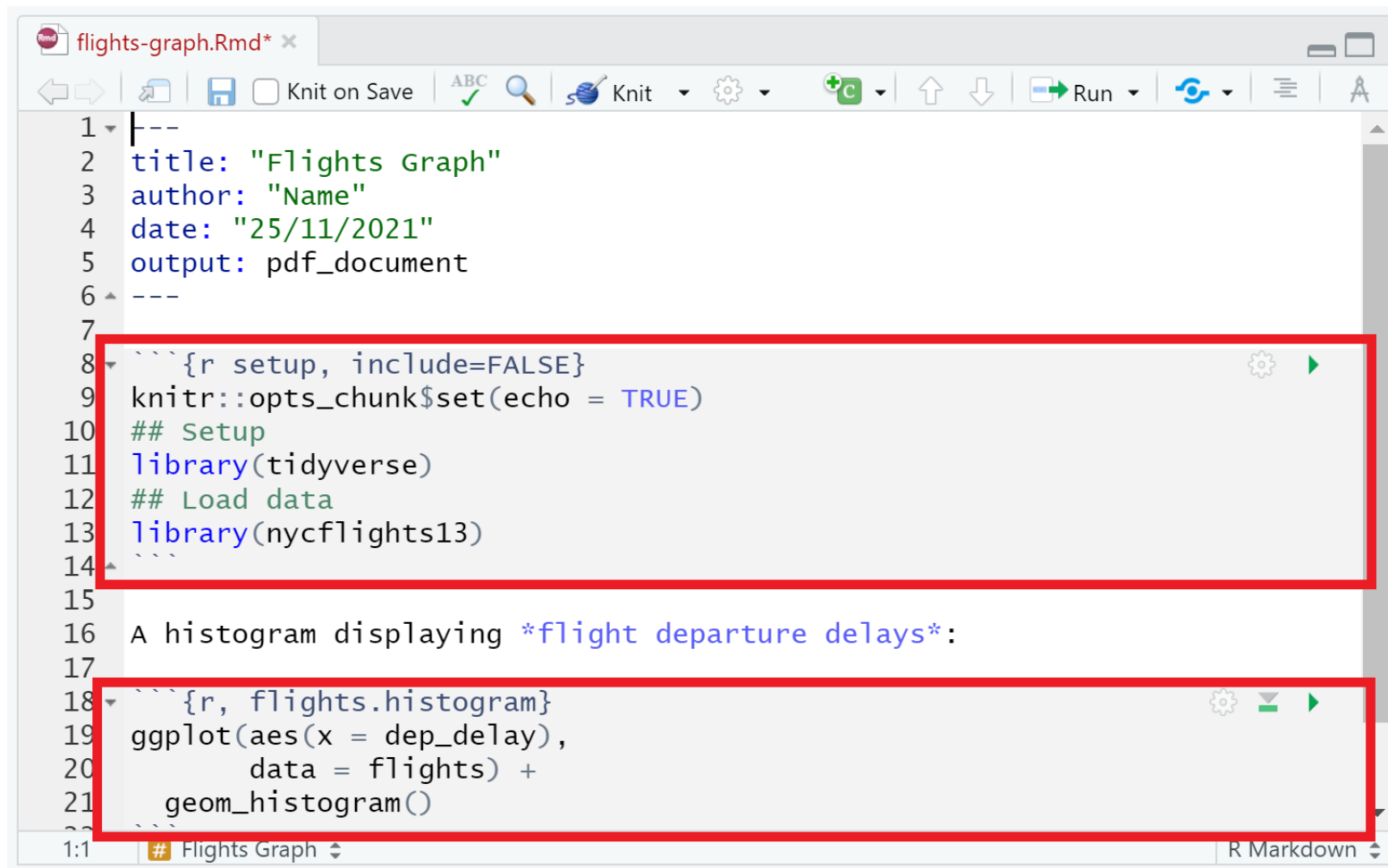


```
1 |  
2 | title: "Flights Graph"  
3 | author: "Name"  
4 | date: "25/11/2021"  
5 | output: pdf_document  
6 | ---  
7 |  
8 | ```{r setup, include=FALSE}  
9 | knitr::opts_chunk$set(echo = TRUE)  
10 | ## Setup  
11 | library(tidyverse)  
12 | ## Load data  
13 | library(nycflights13)  
14 | ```  
15 |  
16 | A histogram displaying *flight departure delays*:  
17 |  
18 | ```{r, flights.histogram}  
19 | ggplot(aes(x = dep_delay),  
20 |         data = flights) +  
21 |   geom_histogram()  
22 | ```
```

1:1 # Flights Graph R Markdown 38 / 45

# Components

You can write in code chunks the same way you would write in a script.



```
1 |  
2 title: "Flights Graph"  
3 author: "Name"  
4 date: "25/11/2021"  
5 output: pdf_document  
6 |  
7 |  
8 {r setup, include=FALSE}  
9 knitr::opts_chunk$set(echo = TRUE)  
10 ## Setup  
11 library(tidyverse)  
12 ## Load data  
13 library(nycflights13)  
14 |  
15 |  
16 A histogram displaying *flight departure delays*:  
17 |  
18 {r, flights.histogram}  
19 ggplot(aes(x = dep_delay),  
20         data = flights) +  
21   geom_histogram()  
22 |  
23 |
```

# Running code

Like with a script, you can use the Run button Cmd/Ctrl + Enter.

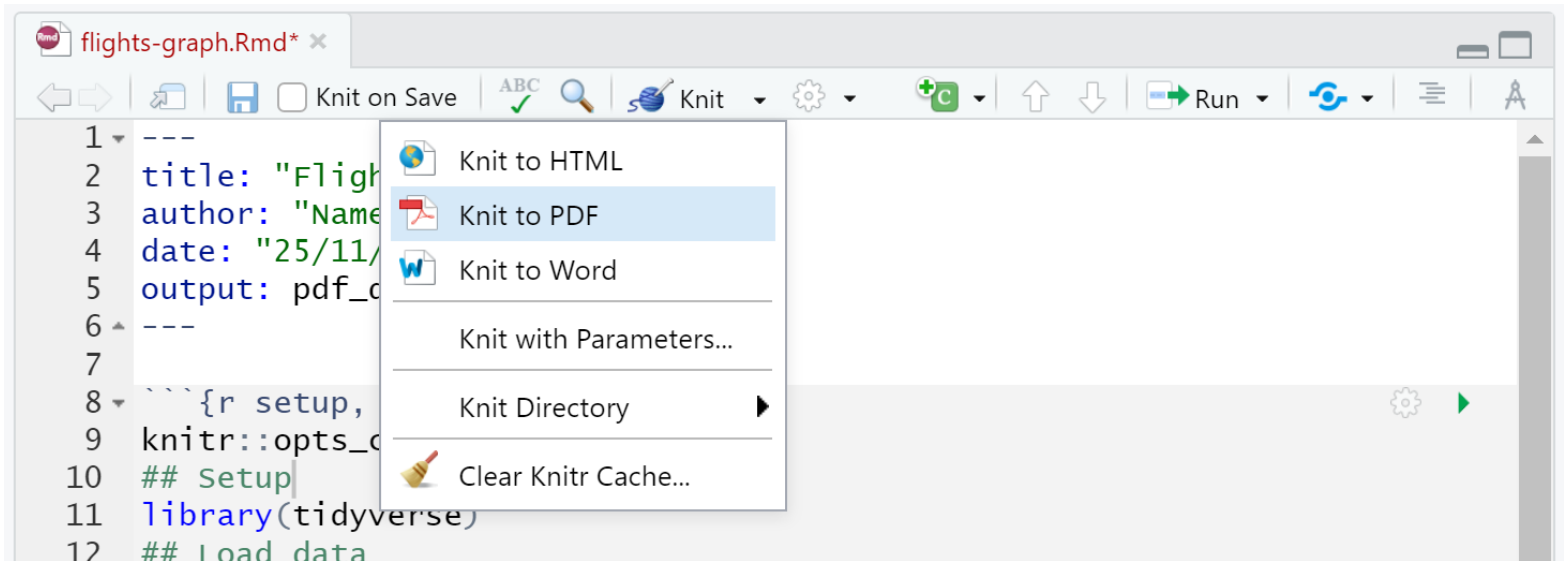
Each chunk also has an arrow you can press to run the code in that chunk.

The output will display below the code chunk rather than in the console or the Plots section.



# Knitting .Rmd files

To present your work, you can knit your R Markdown file to a more common file type, including PDFs, Word documents, and html files.



# Exercises

# R script

1. Create a new R script.
2. Save the script.
3. Write code to calculate the average of ten numbers.
4. Run the script.

# R Markdown

1. Create a new R Markdown file.
2. Save the file.
3. Load the `tidyverse` and `faraway` libraries.
4. Load the dataset "broccoli" by calling the function `data()`
5. Print out the `broccoli` dataset in the R Markdown file.
6. Knit the R Markdown file to PDF.

**Any questions?**