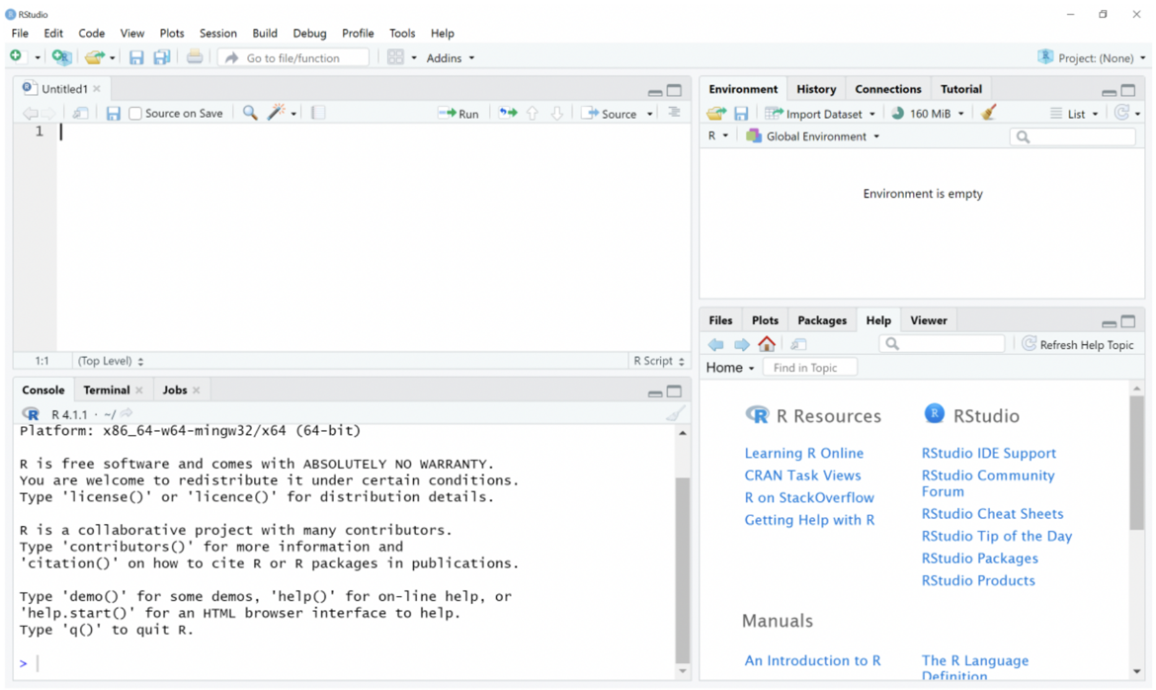
Summary Sheet

AUTHOR
Julia Gallucci

Class 1

Main components of RStudio

1. **Console**: panel where you can execute R code and see the results immediately
2. **Script**: panel where you write and edit your R code. It supports features like syntax highlighting, code completion, and automatic indentation, making it easier to write and read code
3. **Environment:** panel that displays available variables and their values, along with data frames in the current session.
4. **Output/Viewer:** panel for displaying plots generated by R, navigating your file system and manage files and directories, and online help documentation for functions and variables.



```
R 4.1.1 · ~/
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.

>
```

R Basics

Coding style

- comment your code so it's easily interpretable (using #)
- when assigning a variable, use <- , not =
- never reassign reserved words/built in functions (i.e., mean)
- Rules for object names:
    1. Must start with a letter

2. Can only contain letters, numbers, underscores, and periods
3. Typical style conventions; camelCase, snake_case

Things to remember

- R is case sensitive (x is not the same as X)
- When indexing, R starts from 1 (as opposed to languages like python that start at 0)

Math operations

At its most basic function, R works as a “fancy” calculator

Basic Math Operators	Operation
x + y	Addition
x - y	Subtraction
x * y	Multiplication
x / y	Division
x ^ y	Exponent
x %% y	Modulus

Built-in functions

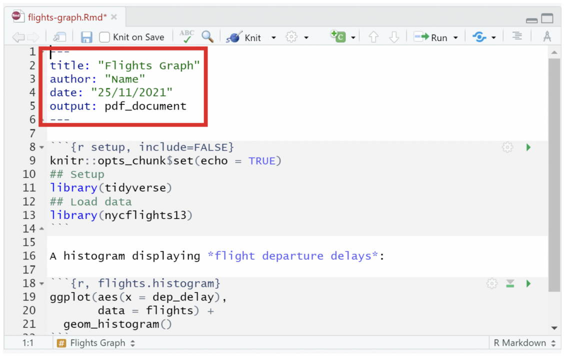
- Packages are collections of R functions, data, and compiled code.
- Libraries are directories in R where the packages are stored.
- Built-in functions are part of R standard or base packages and do not need to be downloaded.
- Typical format:
- `function_name(argument1 = value1, argument2 = value2, …)`
- to find out more information with regards to a package , use `help(function_name)` or `?function_name`
- to install a package that is not built-in to R, use the following commands:

  `install.packages(package_name) to download a package`

  `library(package_name) to load it into your RStudio session`

Main components of RMarkdown

- **YAML header:** contains the document information and settings are specified

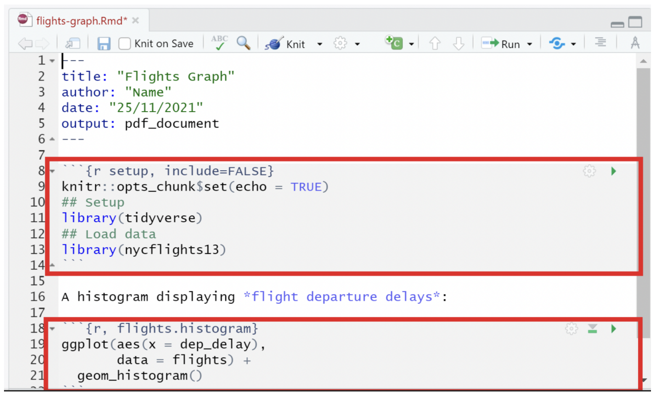


~~~
---
title: "Flights Graph"
author: "Name"
date: "25/11/2021"
output: pdf_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
## Setup
library(tidyverse)
## Load data
library(nycflights13)
```

A histogram displaying *flight departure delays*:

```{r, flights.histogram}
ggplot(aes(x = dep_delay),
       data = flights) +
  geom_histogram()
~~~

- **Chunks**: where code is written. You can write in code chunks the same way you would write in a script

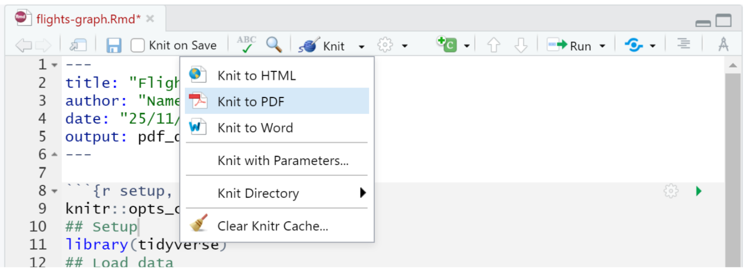
  format of a chunk

  ~~~
  ```{r}  to open a chunk

  ```     to close a chunk
  ~~~



~~~
---
title: "Flights Graph"
author: "Name"
date: "25/11/2021"
output: pdf_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
## Setup
library(tidyverse)
## Load data
library(nycflights13)
```

A histogram displaying *flight departure delays*:

```{r, flights.histogram}
ggplot(aes(x = dep_delay),
       data = flights) +
  geom_histogram()
~~~

  you can knit your RMarkdown file to to a more common file type, including PDFs, Word documents, and html files



~~~
---
title: "Fligh
author: "Name
date: "25/11/
output: pdf_c
---

```{r setup,
knitr::opts_c
## Setup
library(tidyverse)
## Load data
~~~

Main components of a “Reprex” (reproducible example)

1. **Environment:** calls for any necessary libraries and information about your R environment that might be relevant

   session

   ```
   sessionInfo() #to get version information about R, the OS and attached or loaded packages.
   ```

`R.Version()$version.string #provides detailed information about the version of R running.`

`RStudio.Version()``$version #provides detailed information about the version of RStudio running.`

2. **Toy data set:** a minimal data set that the code can be run on. i.e., if you have a large data set, you can select a subset of it and attach that with your reprex.

3. **Code:** minimal and runnable code that recreates the error

```
library(reprex)

reprex({

#code that is producing the error

})
```

Best coding practices:

- Well-comment your code (`using #`)
- Name your variables so they are meaningful and descriptive (`i.e., avoid using x)`
  - To separate words when naming your variable use camelCase or snake_case `(i.e., subjectAge or subject_age)`
  - avoid using reserved words or built-in functions like `mean, TRUE, NA, FALSE etc.`
- code for human readibility (logical spaces and lines)