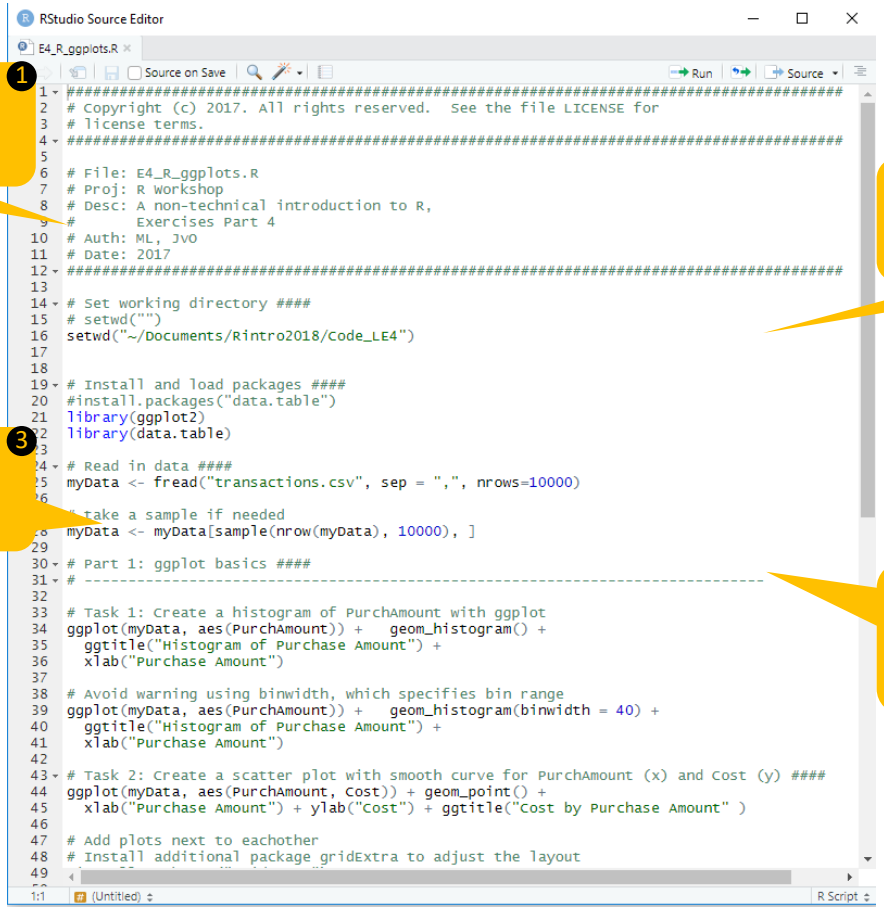
Understanding and creating code notebooks

Extensive data science with R scripts become challenging quickly

1 Comments in your script tend to be terse or non-existent




```
################################################################################
# Copyright (c) 2017. All rights reserved.  See the file LICENSE for
# license terms.
################################################################################

# File: E4_R_ggplots.R
# Proj: R Workshop
# Desc: A non-technical introduction to R,
#       Exercises Part 4
# Auth: ML, JvO
# Date: 2017
################################################################################

# Set working directory ####
# setwd("")
setwd("~/Documents/Rintro2018/Code_LE4")


# Install and load packages ####
#install.packages("data.table")
library(ggplot2)
library(data.table)

# Read in data ####
myData <- fread("transactions.csv", sep = ",", nrows=10000)

# Take a sample if needed
myData <- myData[sample(nrow(myData), 10000), ]

# Part 1: ggplot basics ####
# -----------------------------------------------------------------------------

# Task 1: Create a histogram of PurchAmount with ggplot
ggplot(myData, aes(PurchAmount)) +   geom_histogram() +
  ggtitle("Histogram of Purchase Amount") +
  xlab("Purchase Amount")

# Avoid warning using binwidth, which specifies bin range
ggplot(myData, aes(PurchAmount)) +   geom_histogram(binwidth = 40) +
  ggtitle("Histogram of Purchase Amount") +
  xlab("Purchase Amount")

# Task 2: Create a scatter plot with smooth curve for PurchAmount (x) and Cost (y) ####
ggplot(myData, aes(PurchAmount, Cost)) + geom_point() +
  xlab("Purchase Amount") + ylab("Cost") + ggtitle("Cost by Purchase Amount" )

# Add plots next to eachother
# Install additional package gridExtra to adjust the layout
```

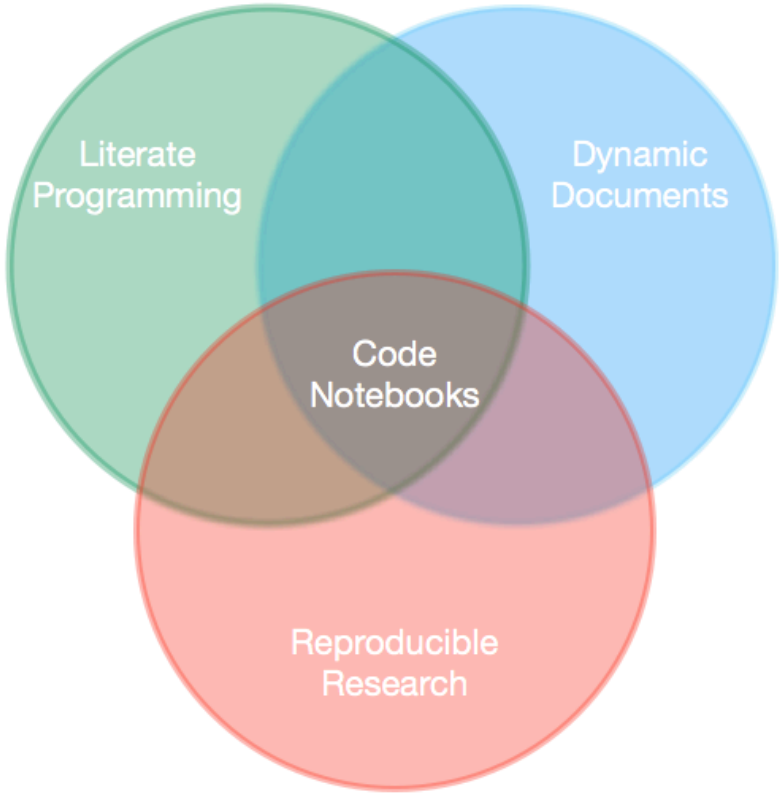
2 Intermediate results are not immediately accessible

3 Coworkers spend a lot of time reading your comments

4 You struggle to find the right plot in your output folder, because there are so many

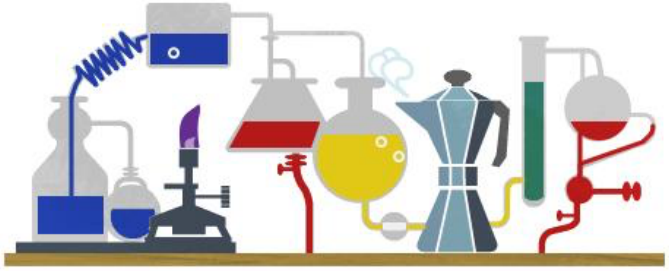
Scripts for data science can become messy quickly.

Code notebooks are a tool to report your code and results and make them publicly available

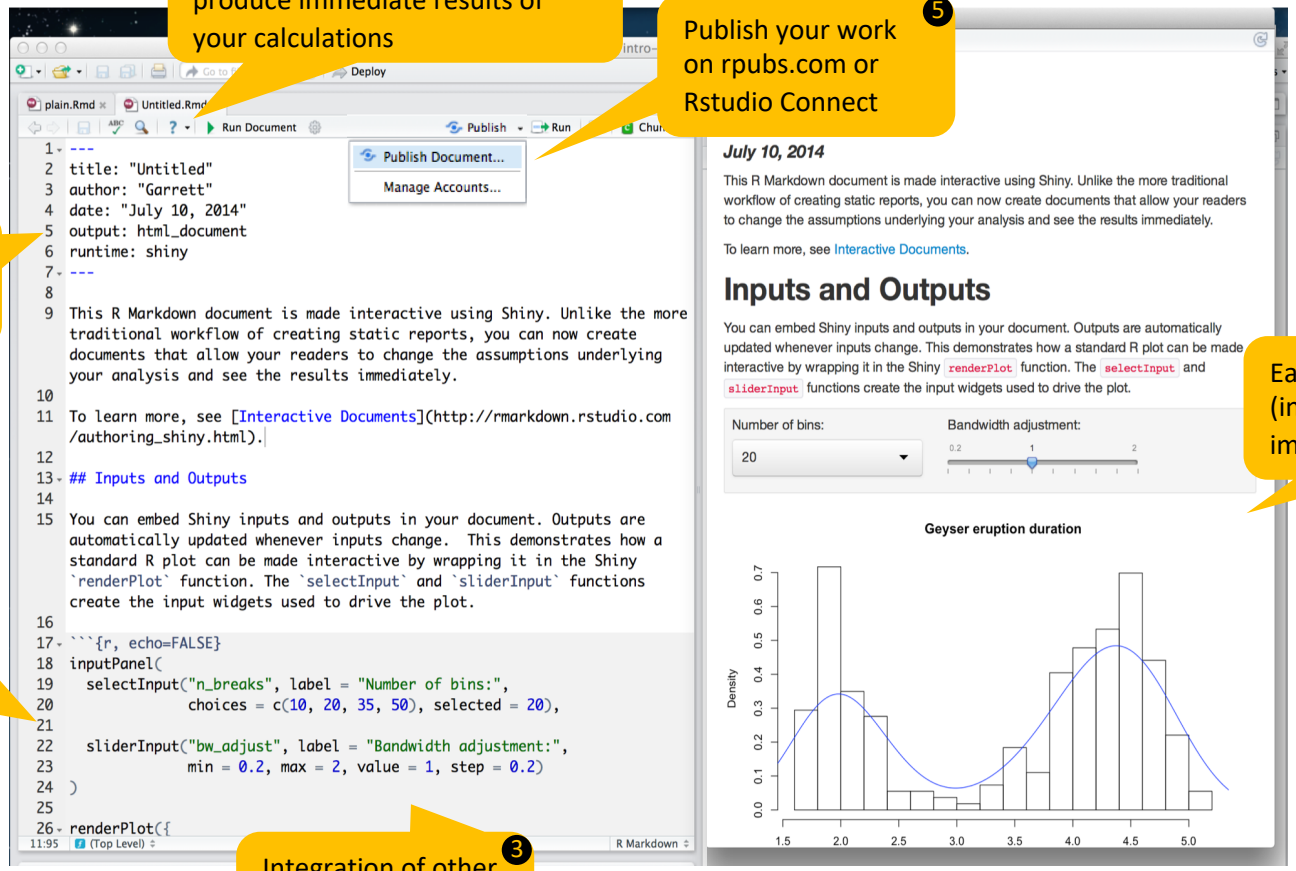


Code notebooks help you to:

- Divide your code into manageable chunks.
  - Execute small chunks of code one by one.
  - Evaluate and store preliminary results.
- Take notes along the process in rich text format (paragraphs, equations, tables, …).
- Present and discuss your results and code chunks with colleagues.



What do you get out of code notebooks?
Example: R Notebook



Notebooks versus regular programming scripts

Activity	Scripts	Notebooks
Building a narrative	**Comments** are available to describe code.	**Rich text** is available to describe the process and conclusion
Manage output	Output is available in the console or environment.	**Output is embedded** in a single document. Code and output may be divided into separate **code chunks**.
Creating a final report	**Creating a report is a separate** and time-consuming **step**.	**Instant report possible**. Reports are publishable as HTML, PDF, …)

1. Allows very descriptive and reproducible documentation
2. Immediate update of output and report
3. Easy sharing possible

Notebooks versus regular programming scripts

Activity	Scripts	Notebooks
Building a narrative	**Comments** are available to describe code.	**Rich text** is available to describe the process and conclusion
Manage output	Output is available in the console or environment.	**Output is embedded** in a single document. Code and output may be divided into separate **code chunks**.
Creating a final report	**Creating a report is a separate** and time-consuming **step**.	**Instant report possible**. Reports are publishable as HTML, PDF, …)

1. Allows very descriptive and reproducible documentation
2. Immediate update of output and report
3. Easy sharing possible

Notebooks versus regular programming scripts

Activity	Scripts	Notebooks
Building a narrative	**Comments** are available to describe code.	**Rich text** is available to describe the process and conclusion
Manage output	Output is available in the console or environment.	**Output is embedded** in a single document. Code and output may be divided into separate **code chunks**.
Creating a final report	**Creating a report is a separate** and time-consuming **step**.	**Instant report possible**. Reports are publishable as HTML, PDF, …)

1. Allows very descriptive and reproducible documentation
2. Immediate update of output and report
3. Easy sharing possible

Notebooks versus regular programming scripts

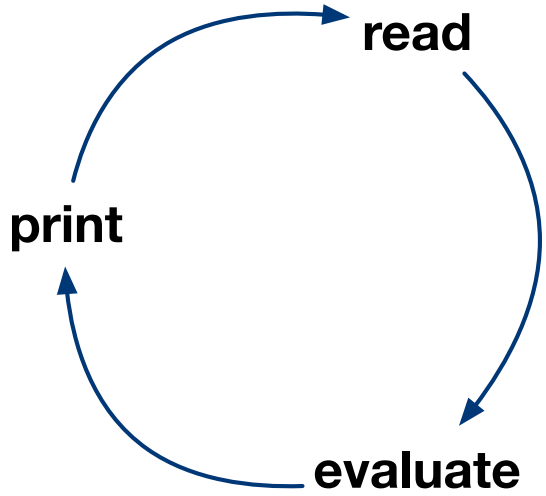
Activity	Scripts	Notebooks
Building a narrative	**Comments** are available to describe code.	**Rich text** is available to describe the process and conclusion
Manage output	Output is available in the console or environment.	**Output is embedded** in a single document. Code and output may be divided into separate **code chunks**.
Creating a final report	**Creating a report is a separate** and time-consuming **step**.	**Instant report possible**. Reports are publishable as HTML, PDF, …)

1. Allows very descriptive and reproducible documentation
2. Immediate update of output and report
3. Easy sharing possible

Notebooks are based on the read-eval-print loop (REPL) technique

The notebook user-interface is developed on the **concept of interactive computing:**

1. User input
2. Input is evaluated
3. Result is printed and notebook is ready for next input



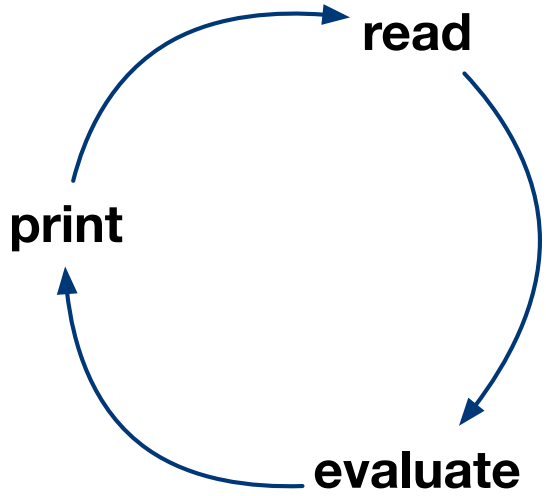
There are **two main features**:

- Store the last output for reuse
- Ability to save input and later reload and re-evaluate

Notebooks are based on the read-eval-print loop (REPL) technique

The notebook user-interface is developed on the **concept of interactive computing:**

1. User input
2. Input is evaluated
3. Result is printed and notebook is ready for next input

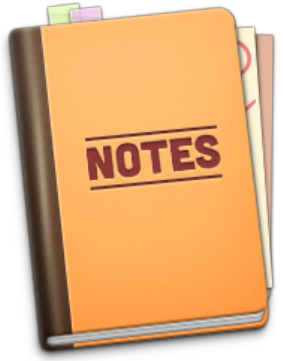


There are **two main features**:

- Store the last output for reuse
- Ability to save input and later reload and re-evaluate

There are many different notebooks, which mostly support multiple programming languages

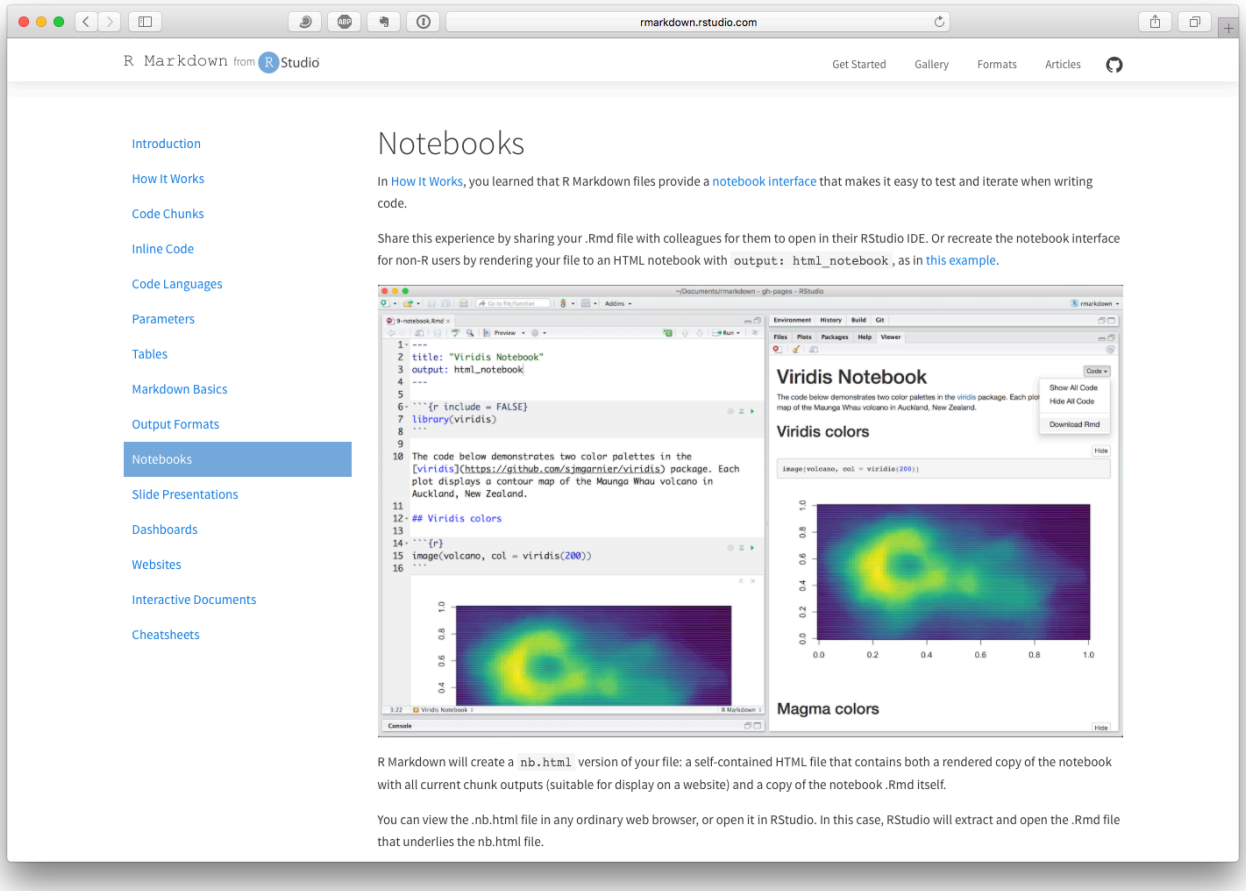
- Jupyter (e.g. used by Google Colab; R, Python; around 40 different languages)
- R Notebooks (R, Python; around 8 different languages)
- Beaker (R, Python; around 17 different languages)
- Zeppelin (R, Python, around 20 different languages)



➢ Which one to pick depends on the requirements of the programming project and your personal preference. We pick to present R notebooks due to its integration into R Studio.

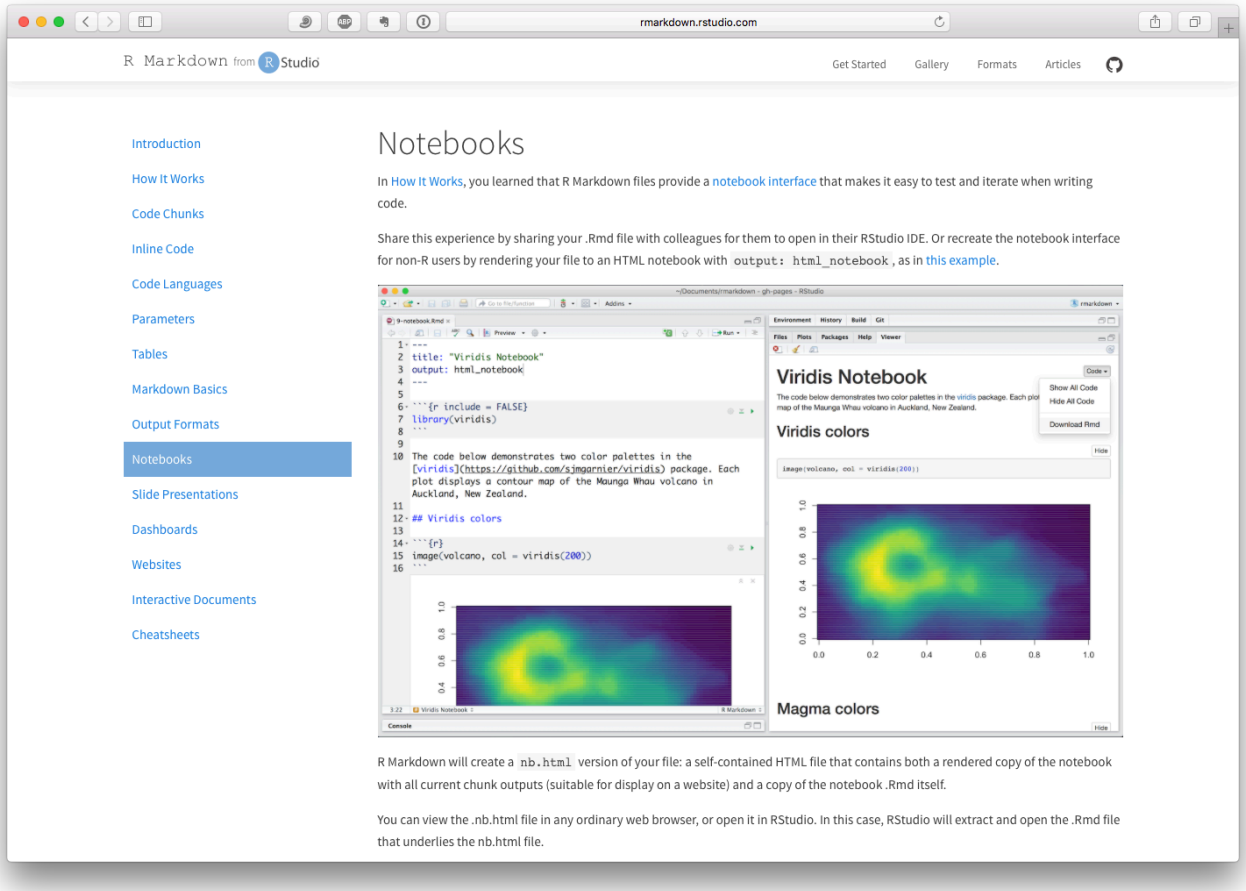
http://beakernotebook.com, https://zeppelin.apache.org, http://jupyter.org, http://jupyter.org

Using R Notebooks in R Studio is very convenient



http://rmarkdown.rstudio.com/lesson-10.html

Using R Notebooks in R Studio is very convenient



http://rmarkdown.rstudio.com/lesson-10.html

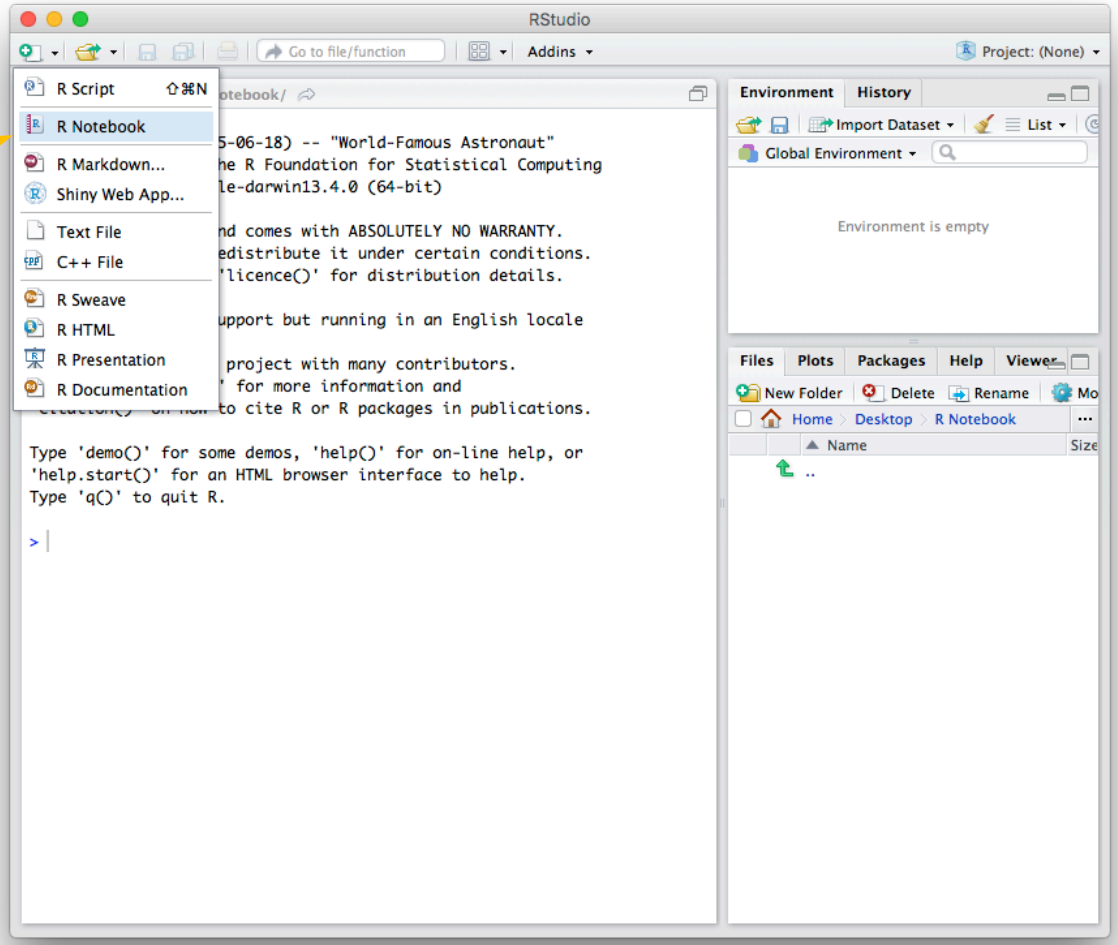
Create your first R Notebook

Steps

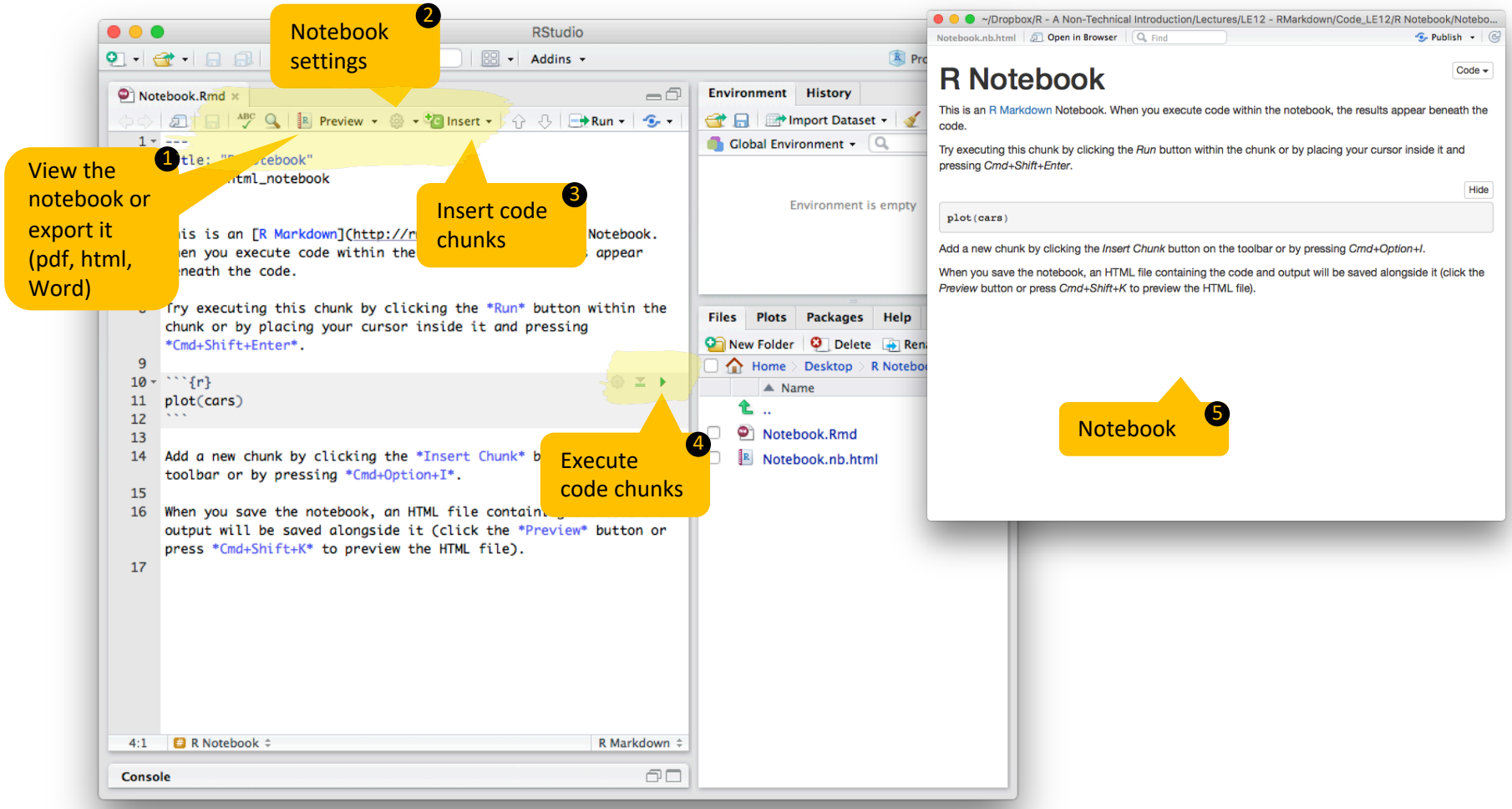
1. **Create a new R Notebook**
2. Create Content
   - Add text elements as Markdown syntax
   - Add code elements in any supported program language
   - Use LaTeX in your R Notebook

Create your first R Notebook

Click on "New" and select "R Notebook"



Main components of an R Notebook



Elements of an R Notebook

There are three types of content in an R Notebook:

- **YAML header (basic information)**

```
1  ---
2  title: "R Notebook"
3  output: html_notebook
4  ---
```

1 The YAML ("Yet another Markup Language") header is optional and surrounded by ---

- **Inline text**

```
11
12 ## R Markdown
13
14 This is an R Markdown document. Markdown is a simple formatting syntax
   for authoring HTML, PDF, and MS Word documents. For more details on using
   R Markdown see <http://rmarkdown.rstudio.com>.
15
```

2 Inline text helps you structuring your document

- **R code chunks**

```
18 ```{r cars}
19 summary(cars)
20 ```
```

3 Code chunks include your R calculations and are surrounded by ```