



Rmarkdown

Creating interactive documents

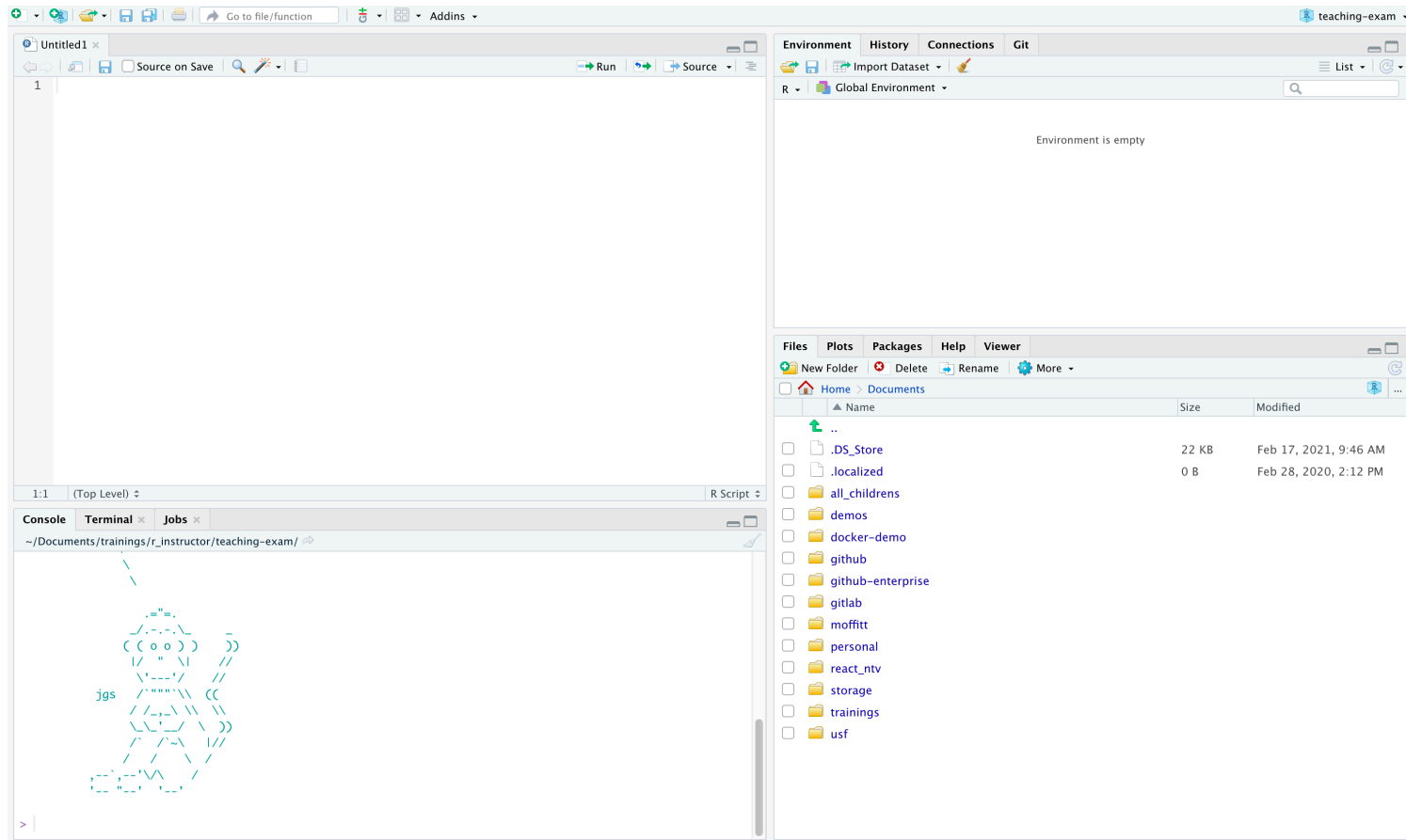
Jordan Creed
Moffitt Cancer Center

June 09, 2021



R & RStudio

RStudio



Packages

A **package** is a collection of functions, data, and documentation that extend the functionality of R.

```
install.packages("foo")
```

vs

```
library("foo")
```

The 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")
```

Data types and structures

Data types:

- logical - `TRUE` or `FALSE`
- numeric or integer - `22.2`
- character - `"character example"`

Data structures:

- vector - `c(1:4, 5, 6)`
- list - `list(c(1:4), 12, "banana")`
- matrix
- dataframe/tibble

`tibble` and `dataframe` - look and perform virtually the same with the biggest difference being that `dataframes` have rows and `tibbles` do not

RStudio IDE : : CHEAT SHEET

Documents and Apps

Annotations for Documents and Apps:

- Open Shiny, R Markdown, knitr, Sweave, LaTeX, Rst files and more in Source Pane
- Check spelling
- Render output
- Choose output format
- Choose output location
- Insert code chunk
- Jump to previous chunk
- Jump to next chunk
- Run selected lines
- Publish to server
- Show file outline
- Access markdown guide at **Help > Markdown Quick Reference**
- Jump to chunk
- Set knitr and chunk options
- Run this code chunk
- Run this code chunk

RStudio recognizes that files named **app.R**, **server.R**, **ui.R**, and **global.R** belong to a shiny app

Run Choose Publish Manage app location to shinyapps.io or server publish accounts

Write Code

Annotations for Write Code:

- Navigate Open in new window Save Find and replace Compile as notebook Run selected code
- Source with or without Echo
- Multiple cursors/column selection with **Alt + mouse drag**
- Code diagnostics that appear in the margin. Hover over diagnostic symbols for details.
- Syntax highlighting based on your file's extension
- Tab completion to finish function names, file paths, arguments, and more.
- Multi-language code snippets to quickly use common blocks of code.
- Change file type
- Working Directory
- Maximize, minimize panes
- Drag pane to see command history

R Support

Annotations for R Support:

- Import data with wizard
- History of past commands to run/copy
- Display R/RPS slideshows **File > New File > R Presentation**
- Load workspace
- Save workspace
- Delete all saved objects
- Search inside environment
- Choose environment to display from list of parent environments
- Display objects as list or grid
- Displays saved objects by type with short description
- View in data viewer
- View function source code
- Create folder
- Upload file
- Delete file
- Rename file
- Export
- Go to Working Directory
- Change directory
- Path to displayed directory
- A file browser keyed to your working directory. Click on file or directory name to open.

Pro Features

Annotations for Pro Features:

- Share Project with Collaborators
- Active shared collaborators
- Start new R Session in current project
- Close R Session in project
- Select R Version
- PROJECT SYSTEM**
- File > New Project**
- RStudio saves the call history, workspace, and working directory associated with a project. It reloads each when you re-open a project.
- Name of current project
- RStudio opens plots in a dedicated Plots pane
- Navigate recent plots
- Open in plot window
- Export plot
- Delete plot
- Delete all plots
- GUI Package manager lists every installed package
- Install Packages
- Update Packages
- Create reproducible package library for your project
- Click to load package with **library()**. Unlick to detach package with **detach()**
- Package version installed
- Delete from library
- RStudio opens documentation in a dedicated Help pane
- Home page of helpful links
- Search within help file
- Search for help file
- Viewer Pane displays HTML content, such as Shiny apps, RMarkdown reports, and interactive visualizations
- Stop Shiny app
- Publish to shinyapps.io, rpubs, RSCconnect
- Refresh
- View(<data>)** opens spreadsheet like view of data set

Debug Mode

Annotations for Debug Mode:

- Open with **debug()**, **browser()**, or a breakpoint. RStudio will open the debugger mode when it encounters a breakpoint while executing code.
- Launch debugger mode from origin of error
- Open traceback to examine the functions that R called before the error occurred
- Click next to line number to add/remove a breakpoint
- Highlighted line shows where execution has paused
- Run commands in environment where execution has paused
- Examine variables in executing environment
- Select function in traceback to debug
- Step through code one line at a time
- Step into and out of functions to run
- Resume execution mode
- Quit debug

Version Control with Git or SVN

Annotations for Version Control:

- Turn on at **Tools > Project Options > Git/SVN**
- Stage files
- Show file diff
- Commit staged files to remote
- Push/Pull to remote
- View History
- Added
- Deleted
- Modified
- Renamed
- Untracked
- Open shell to type commands
- current branch

Package Writing

Annotations for Package Writing:

- File > New Project > New Directory > R Package**
- Turn project into package.
- Enable roxygen documentation with **Tools > Project Options > Build Tools**
- Roxygen guide at **Help > Roxygen Quick Reference**



RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at www.rstudio.com • RStudio IDE 0.99.832 • Updated: 2016-01

r cheatsheets



R Projects & `here()`

Working directory = where R looks for your files

Setting absolute paths can be problematic - especially if you are working on a project that uses Windows/Mac/Linux

R Projects provide a structure for storing all files, data, scripts and output for a project

To create a new Project: **File** > **New Directory** > **New Project** and fill out the information

☁ In RStudio Cloud we will be working inside a project so you will not have the option to create projects and switch between them.

Rmarkdown

Rmarkdown

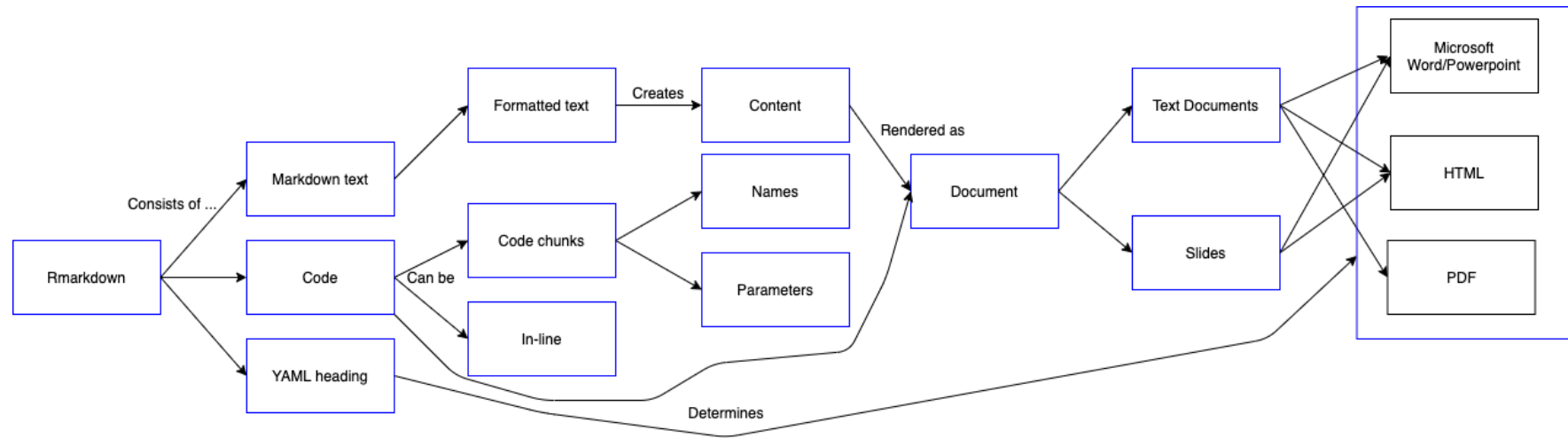


Allows you to share code, comments and outputs easily

☆ REPRODUCIBLE! ☆

3 elements: YAML header, formatted text, code

Rmarkdown



YAML

Located at the top of your file and enclosed by `---`

Controls "whole document" parameters and settings

```
---  
title: "My super cool document"  
author:  
  - Jordan Creed  
output: html_document  
---
```

Formatted Text

Rmarkdown uses regular markdown rules/syntax.

When you `knit` your document the final output will display according to the formatting rules on your `.Rmd`

Code

Code can be written as either code chunks or inline code

Output can be modified by chunk options placed inside `{ }`

- `eval = FALSE` keeps code from being evaluated
- `echo = FALSE` keeps code from being displayed
- `message = FALSE` and `warning = FALSE` keeps messages/warnings from being displayed

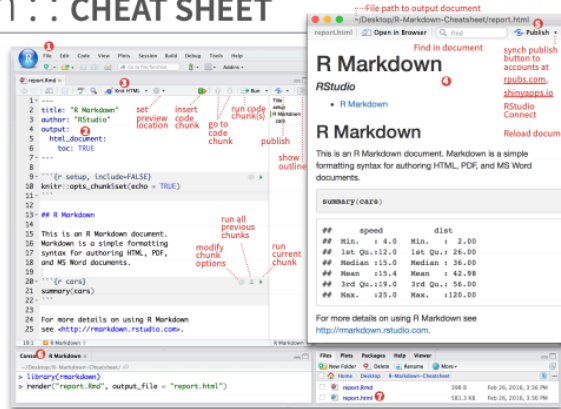
R Markdown :: CHEAT SHEET

What is R Markdown?

- .Rmd files** - An R Markdown (.Rmd) file is a record of your research. It contains the code that a scientist needs to reproduce your work along with the narration that a reader needs to understand your work.
- Reproducible Research** - At the click of a button, or the type of a command, you can rerun the code in an R Markdown file to reproduce your work and export the results as a finished report.
- Dynamic Documents** - You can choose to export the finished report in a variety of formats, including HTML, PDF, MS Word, or RTF documents, HTML or pdf based slides, Notebooks, and more.

Workflow

1. **Open a new .Rmd file** at File > New File > R Markdown. Use the wizard that opens to pre-populate the file with a template
2. **Write document** by editing template
3. **Knit document to create report**; use knit button or `render()` to knit
4. **Preview Output** in IDE window
5. **Publish** (optional) to web server
6. **Examine build log** in R Markdown console
7. **Use output file** that is saved along side .Rmd



render

Use `rmarkdown::render()` to render/knit at cmd line. Important args:

input	output_options	output_file	params	envir	encoding
file to render	List of render options (as in YAML)	output_dir	list of params to use	environment to evaluate code chunks in	of input file

Embed code with knitr syntax

INLINE CODE
Insert with `<code>`. Results appear as text without code.
Built with `<code>` Built with 3.2.3

CODE CHUNKS
One or more lines surrounded with `<code>` and `<code>`. Place chunk options within curly braces, after `<code>`. Insert with `<code>`

GLOBAL OPTIONS
Set with `<code>`, e.g.
`<code>`
`<code>`

IMPORTANT CHUNK OPTIONS

- cache** - cache results for future knits (default = FALSE)
- cache.path** - directory to save cached results in (default = "cache")
- child** - file(s) to knit and then include (default = NULL)
- collapse** - collapse all output into single block (default = FALSE)
- comment** - prefix for each line of results (default = "#")

- dependson** - chunk dependencies for caching (default = NULL)
- echo** - Display code in output document (default = TRUE)
- engine** - code language used in chunk (default = "R")
- error** - Display error messages in doc (TRUE) or stop render when errors occur (FALSE) (default = FALSE)
- eval** - Run code in chunk (default = TRUE)

- fig.align** - "left", "right", or "center" (default = "center")
- fig.cap** - figure caption as character string (default = NULL)
- fig.height**, **fig.width** - Dimensions of plots in inches
- highlight** - highlight source code (default = TRUE)
- include** - include chunk in doc after running (default = TRUE)
- message** - display code messages in document (default = TRUE)
- results** (default = "markup")
"asis" - pass through results
"hide" - do not display results
"hold" - put all results below all code
- tidy** - tidy code for display (default = FALSE)
- warning** - display code warnings in document (default = TRUE)

Options not listed above: R.options, autoexec, background, cache.comments, cache.lazy, cache.rebuild, cache.vars, dev, dev.args, dpl, engine.opts, engine.path, fig.asp, fig.env, fig.keep, fig.la, fig.path, fig.pos, fig.process, fig.retina, fig.scap, fig.show, fig.showtext, fig.subcap, interval, out.extra, out.height, out.width, prompt, purr, rel.label, render, size, split, tidy.opts



RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at rmarkdown.rstudio.com • rmarkdown 1.6 • Updated: 2016-02

.rmd Structure

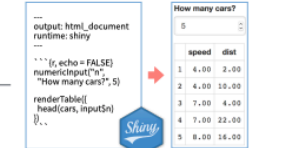
YAML Header
Optional section of render (e.g. pandoc) options written as keyvalue pairs (YAML).
At start of file
Between lines of ---
Text
Narration formatted with markdown, mixed with:
Code Chunks
Chunks of embedded code. Each chunk:
Begins with `<code>`
ends with `<code>`
R Markdown will run the code and append the results to the doc.
It will use the location of the .Rmd file as the **working directory**

Parameters

- Parameterize your documents to reuse with new inputs (e.g., data, values, etc.)
1. **Add parameters** - Create and set parameters in the header as sub-values of params
 2. **Call parameters** - Call parameter values in code as `params$<name>`
 3. **Set parameters** - Set values with Knit with parameters or the params argument of render():
`render("doc.Rmd", params = list(n = 1, d = as.Date("2015-01-01")))`

Interactive Documents

- Turn your report into an interactive Shiny document in 4 steps
1. Add runtime: shiny to the YAML header.
 2. Call Shiny input functions to embed input objects.
 3. Call Shiny render functions to embed reactive output.
 4. Render `w rmarkdown::run` or click Run Document in RStudio IDE



Embed a complete app into your document with `shiny::shinyAppDir()`

Publish on RStudio Connect, to share R Markdown documents securely, schedule automatic updates, and interact with parameters in real time.
www.rstudio.com/products/connect/

GitHub

- GitHub is a code hosting platform used for collaborating and code sharing
- Materials and information for the course can be found on the class GitHub page
- Class GitHub:
<https://github.com/FridleyLab/Introduction-to-R>



RStudio Cloud

To get started:

- Create an account with **RStudio Cloud**
- Log in
- To your right you should see "Introduction To R Programming Class (2021)" - click on it
- Under "Projects" press "Start" beside "Lab One"