

# Using Rmarkdown to generate dynamic documents

AECN 396/896-002

# Table of contents

1. What is Rmarkdown?
2. Chunk options
3. Caching
4. YAML
5. Directory
6. Output types

# What is Rmarkdown?

---

# What is and why Rmarkdown?

- It allows you to effortlessly generate documents (or even websites) that can print both R codes and their outcomes (this lecture note is indeed written using **Rmarkdown**) in a single document.
- The full power of Rmarkdown is on display [here](#).
- It is useful when you report the analysis you conducted and its source R codes to your advisor or anyone you report to (as long as that person understands R).

The power of Rmarkdown goes well beyond just creating a simple html document.

- book
- interactive map
- dashboard
- website
- presentation slides (this lecture is made using Rmarkdown)

# Using WORD?

- It would be a real pain to do so because you need to copy and paste all the R codes you run and the results onto WORD **manually**.
- Often times, copied R codes and results are very much likely to be badly formatted when pasting them
- Rmarkdown obviates the need of repeating copying and pasting when you would like to communicate what you did (R codes) and what you found (results) without worrying too much about formatting.

# Generating a report using Rmarkdown

Generating a report using Rmarkdown is a two-step process:

1. Create an Rmarkdown file (file with .Rmd as an extension) with regular texts and R codes mixed inside it.
  - You use a special syntax to let the computer know which parts of the file are simple texts and which parts are R codes.
2. Tell the computer to process the Rmd file (a click of a button on Rstudio, or use the `render()` function)
  - The computer runs the R codes and get their outcomes
  - Combine the text parts, R codes, and their results to produce a document

# Rmarkdown: the Basics

---



# R code chunks

Rmd file consists of two types of inputs:

- R code chunks
- Regular texts

## special syntax

We can indicate R codes chunks by placing R codes inside a special syntax.

```
```${r}  
codes here  
```
```

### Direction

Take a look at `rmd_sample.Rmd`, which can be downloaded from [here](#).

- R codes `summary(cars)` and `plot(pressure)` are enclosed individually by the special syntax
- So, in this rmd file, R knows that it should treat them as R codes, but not regular texts.
- On the other hand, any texts that are not enclosed by the special syntax would be recognized as regular text.

# Let's knit

The process of compiling an rmd file to produce a document is called **knitting**.

- The easiest way to knit is to hit the Knit button located at the top of the code pane (upper left pane by default)
- Alternatively, you can use the `render()` function to knit like below:

```
render("sample_rmd.Rmd")
```

### Direction

Inspect the rmd file and its output document:

### Rmd side

1. lines 1-15: a **YAML header** where you control the aesthetics of the output document (more on this later)
2. lines 36-38: texts not enclosed by the special syntax
3. lines 40-42: `summary(cars)` is an R code, which is enclosed by the special syntax

### html side

1. lines 1-15: nothing
2. lines 36-38: printed as regular texts
3. lines 40-42: the R code and its results printed

# Inline code

You can refer to an R object previously defined in line and display its content in line:

## Direction

See lines 62-68 of the rmd file

# Markdown basics

- header
- make a list
- font
- code highlighting
- inline math
- math
- web link
- citation

## Direction

Compare Chapter 2 of the rmd file and its output html file

# Caveat when knitting

- When you knit an rmarkdown file and create a report, R creates an R session/environment that is completely independent of whatever R sessions or environments you may have on your RStudio.
- This means that when you knit an rmarkdown file, you cannot refer to R objects you have defined on your current R session.

# Chunk options

---



## Direction

Inspect the rmd file and its output document

- From the R code chunk with `summary(cars)`, the code itself and its outcome are presented in the output
- From the R code chunk with `plot(pressure)`, only its outcome is presented in the output

## chunk options: echo

- This is because of the chunk option `echo = FALSE` in the second R code chunk
- We can control how R code chunks are translated and appear in the output document using chunk options

### Various chunk options

- `echo`
- `eval`
- `message`
- `warning`
- `results`
- `include`
- `cache`
- `fig.cap`

### Chunk option: echo

- `echo` ( `TRUE` or FALSE): specify whether the R codes appear in the output document or not
- `eval` ( `TRUE` or FALSE): specify whether the R codes are evaluated or not

### Direction

Inspect the rmd file (lines 95-115) and its output document.

<sup>1</sup> Values in red indicates that they are the default values

### Chunk option: message

- `message` (TRUE or FALSE): specify whether messages associated with R code evaluation appear in the output document or not
- `warning` (TRUE or FALSE): specify whether warnings associated with R code evaluation appear in the output document or not

### Direction

Inspect the rmd file (lines 119-140) and its output document

### Chunk option: results

`results` ( `markup` , `hide`, `asis`)

- `hide`: hides all the results including warnings and messages
- `asis`: the outputs of the R codes are printed as-is without any suitable formatting (which the default option `markup` does)

### Direction

Inspect the rmd file (lines 144-156) and its output document

<sup>1</sup> Values in red indicates that they are the default values

<sup>2</sup> `asis` is useful when using the `stargazer()` function to present regression results.

### Chunk option: include

`include = FALSE` is equivalent to having `eval = TRUE, echo = FALSE, results = "hide"`

### Direction

Inspect the rmd file (lines 160-173) and its output document

# Specify chunk options globally

Sometimes, it is useful to set chunk options that apply globally (for the entire documents).

For example,

- You are writing a term paper and the instructor may want to see only results, but not R codes.
- You do not want any of the R codes to appear on the output document, but `echo = TRUE` is the default.

# Specify chunk options globally

You can use `opts_chunk$set()` from the **knitr** package to set chunk options globally.

```
opts_chunk$set(  
  echo = FALSE,  
  messages = FALSE,  
  warnings = FALSE,  
  fig.align = "center",  
  fig.width = 5,  
  fig.height = 4  
)
```

## Direction

Remove `eval = F` from the R chunk at line 17 and knit.

**Important**: Local option overrides the global option.



### Chunk option for figure

- `fig.align`: 'default', 'center', 'left', 'right'
- `fig.width`: in inches
- `fig.height`: in inches
- `fig.cap`: figure caption

### Direction

Play with these options. See [here](#) for more chunk options.

# Caching

---

- In the course of creating a document using Rmarkdown, You are going to hit the "Knit" button numerous times when you are writing a report to check whether the final output looks fine.
- Every time you knit, all the R code chunks are evaluated, which is inefficient because R has evaluated those R code chunks before.
- So, if we can somehow store the results of R code chunks (caching), and then let R call up the saved results instead of re-evaluating the codes all over again, we can save lots of time.
- The benefit of doing so is greater when the processing time of the codes is longer. Caching can be done by adding `cache==TRUE` as a chunk option.
- By adding the option, once an R chunk is processed, its results are saved and can be reused again by R later when you compile the document again.

### Direction

- change `eval = F` to `eval = T` in the **cache\_1** chunk
- knit and confirm that **sample\_rmd\_cache** and **sample\_rmd\_files** folders are created
- knit again and observe that the knitting process is much faster now

When any part of the R codes within a cached R code chunk is changed, R is smart enough to recognize the change and evaluate the R code chunk again without using the cached results for the chunk.

#### Direction

Change `y = 1 + x + v` to `y = 1 + 2 * x + v` in the **cache\_1** chunk and knit

- Sometimes, your R codes within an cached R code chunk have not changed, but the content of a dataset used in the R code chunk may have changed.
- In such a case, R is unable to recognize the change in the content of the dataset.

#### Direction

- change `eval = F` to `eval = T` in the `cache_2` chunk and knit
- change `y = 1 + 2 * x + v` back to `y = 1 + x + v` and knit (notice that the printed number from `cache_2` did not change)

- To R, everything in the `cache_2` chunk looks the same as they only look at the code texts, but not the contents of R objects.
- Therefore, R would call up the saved results instead of rerunning the R codes, which is not what you want.
- You can use the `dependson` option to make R recognize any changes in cached R objects

#### Direction

Add `dependson = "cache_1"` to the `cache_2` chunk as an option and knit again.

# YAML

---



# YAML

At the very beginning of an rmd file, you have a YAML header, where you specify how the resulting documents look like.

## Example

```
---  
title: "Reporting using Rmarkdown"  
author: "Taro Mieno"  
date: "2022-08-24"  
output:  
  html_document:  
    number_sections: yes  
    theme: flatly  
    toc: yes  
    toc_float: yes  
  word_document:  
    toc: yes  
---
```

## Direction

- Try other themes and highlight methods found [here](#)
- Visit [here](#) for the other options.

# Directory

---

# Reading Files

Suppose you are interested in reading a dataset file like this:

```
read.csv("corn_price.csv")
```

## Important:

By default, RStudio looks for **corn\_price.csv** in the same folder in which the Rmd file is located.

## So,

- In my case, the sample\_rmd.rmd is located in /Users/tmieno2/Dropbox/TeachingUNL/Data-Science-with-R/Chapter-2-Rmarkdown.
- So, RStudio tries to find /Users/tmieno2/Dropbox/TeachingUNL/Data-Science-with-R/Chapter-2-Rmarkdown/corn\_price.csv.
- If the file is not in the directory, RStudio won't be able to find the file to import and returns an error. Clearly, all the subsequent actions dependent on the dataset will not run.

To avoid errors in reading files, there are three options:

**Option 1.** (recommended for a beginner)

Put all the datasets you intend to use in the same directory in which your rmd file is located

**Option 2.**

If the file is not in the directory, supply the full path to the file like this

```
read.csv("~/Dropbox/TeachingUNL/Data-Science-with-R/Chapter-2-Rmarkdown/corn_price.csv")
```

**Option 3.**

Tell R to look for a specific directory for datasets by setting a working directory using `opts_knit$set(root.dir = directory)` at the beginning

```
opts_knit$set(root.dir = "~/Dropbox/TeachingUNL/Data-Science-with-R/Chapter-2-Rmarkdown/")
```

# Output types

---

## Output types

- html
- Word
- pdf

## How

To select the output type, first click on the black triangle button next to the "Knit" button, and then select your preferred option type.

- By far the preferred version of the three is html if you do not intend to print out the output document.
- html is void of the concept of **page**. Consequently, you do not have to worry about how you should organize texts, tables, and figures within a page (fixed amount of space).
- The formatting of figure and tables can be screwed up in Word

# Interactive features

html is much more powerful than pdf or WORD in the sense that it is interactive.

**htmlwidgets**: tools to generate interactive contents using R

See the list of htmlwidgets [here](#)

# Interactive features: table

`datatable()` from the `DT` package lets you create an interactive table

```
library(tidyverse)
library(DT)

iris %>% datatable(
  extensions = "Buttons",
  options = list(
    dom = "Blfrtip",
    buttons = c("copy", "csv", "excel", "pdf", "print"),
    lengthMenu = list(
      c(10, 25, 50, -1),
      c(10, 25, 50, "All")
    )
  )
)
```



# Interactive features: table

Copy

CSV

Excel

PDF

Print

Show

10

▼

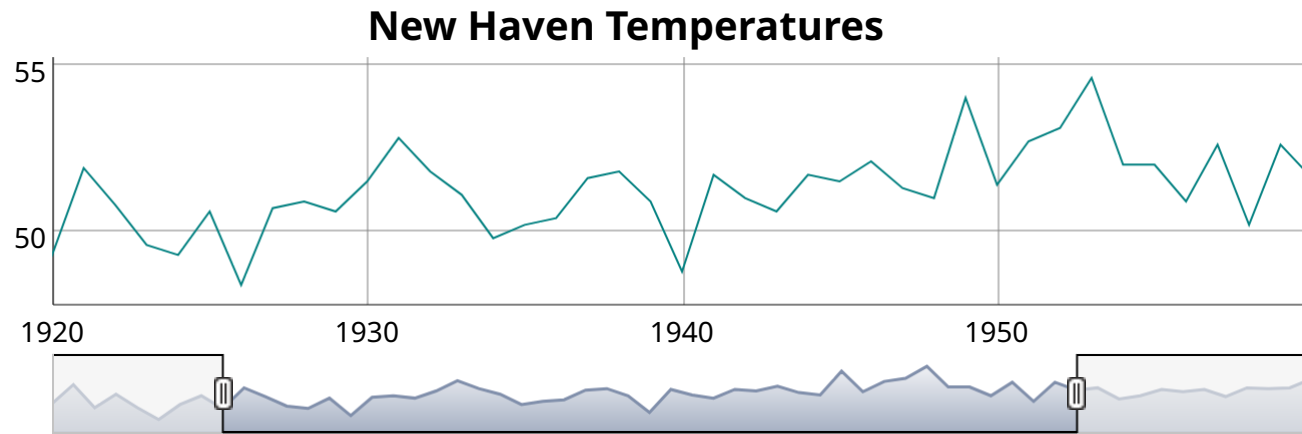
entries

Search:

|    | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|----|--------------|-------------|--------------|-------------|---------|
| 1  | 5.1          | 3.5         | 1.4          | 0.2         | setosa  |
| 2  | 4.9          | 3           | 1.4          | 0.2         | setosa  |
| 3  | 4.7          | 3.2         | 1.3          | 0.2         | setosa  |
| 4  | 4.6          | 3.1         | 1.5          | 0.2         | setosa  |
| 5  | 5            | 3.6         | 1.4          | 0.2         | setosa  |
| 6  | 5.4          | 3.9         | 1.7          | 0.4         | setosa  |
| 7  | 4.6          | 3.4         | 1.4          | 0.3         | setosa  |
| 8  | 5            | 3.4         | 1.5          | 0.2         | setosa  |
| 9  | 4.4          | 2.9         | 1.4          | 0.2         | setosa  |
| 10 | 4.9          | 3.1         | 1.5          | 0.1         | setosa  |

# Interactive features: time-series data

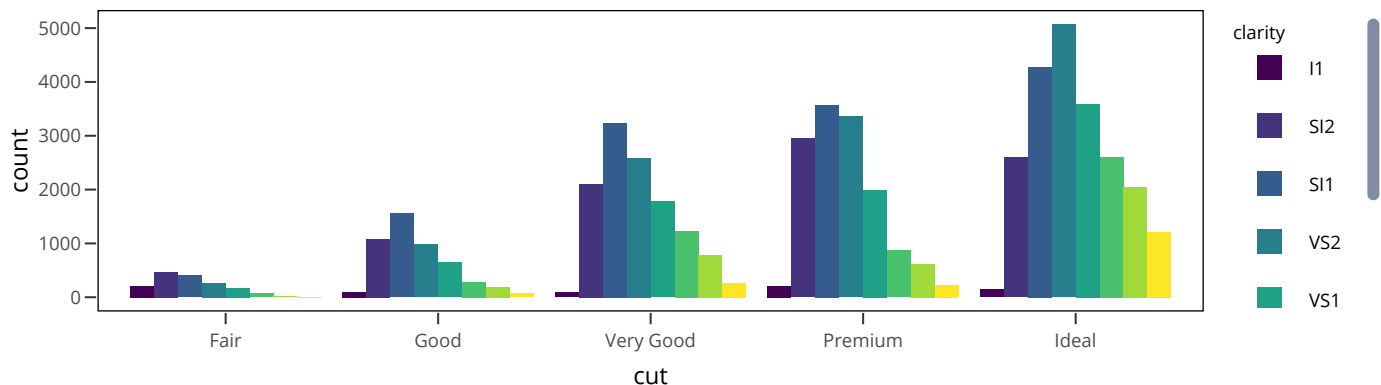
```
library(dygraphs)
dygraph(nhtemp, main = "New Haven Temperatures") %>%
  dyRangeSelector(dateWindow = c("1920-01-01", "1960-01-01"))
```



# Interactive features: ggplot2 figures

```
library(ggplot2)
library(plotly)
p <- ggplot(data = diamonds) +
  geom_bar(aes(x = cut, fill = clarity), position = "dodge")

ggplotly(p)
```



# Resources

Here is a list of some useful resources to learn Rmarkdown.

- [R Markdown: The Definitive Guide](#)
- [Introduction to R markdown by Rstudio](#)
- [Cheat sheet by Rstudio](#)