

# R Markdown Cheat Sheet

learn more at [rmarkdown.rstudio.com](http://rmarkdown.rstudio.com)



## .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 as a html, pdf, MS Word, ODT, RTF, or markdown document; or as a html or pdf based slide show.

## 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 or server

Synch publish button to accounts at

- [rpubs.com](http://rpubs.com),
- [shinyapps.io](http://shinyapps.io)
- RStudio Connect

Reload document  
Find in document  
File path to output document

- 6 **Examine build log** in R Markdown console

- 7 **Use output file** that is saved alongside .Rmd

## .Rmd structure

### YAML Header

Optional section of render (e.g. pandoc) options written as key:value pairs (YAML).

- At start of file
- Between lines of ---

### Text

Narration formatted with markdown, mixed with:

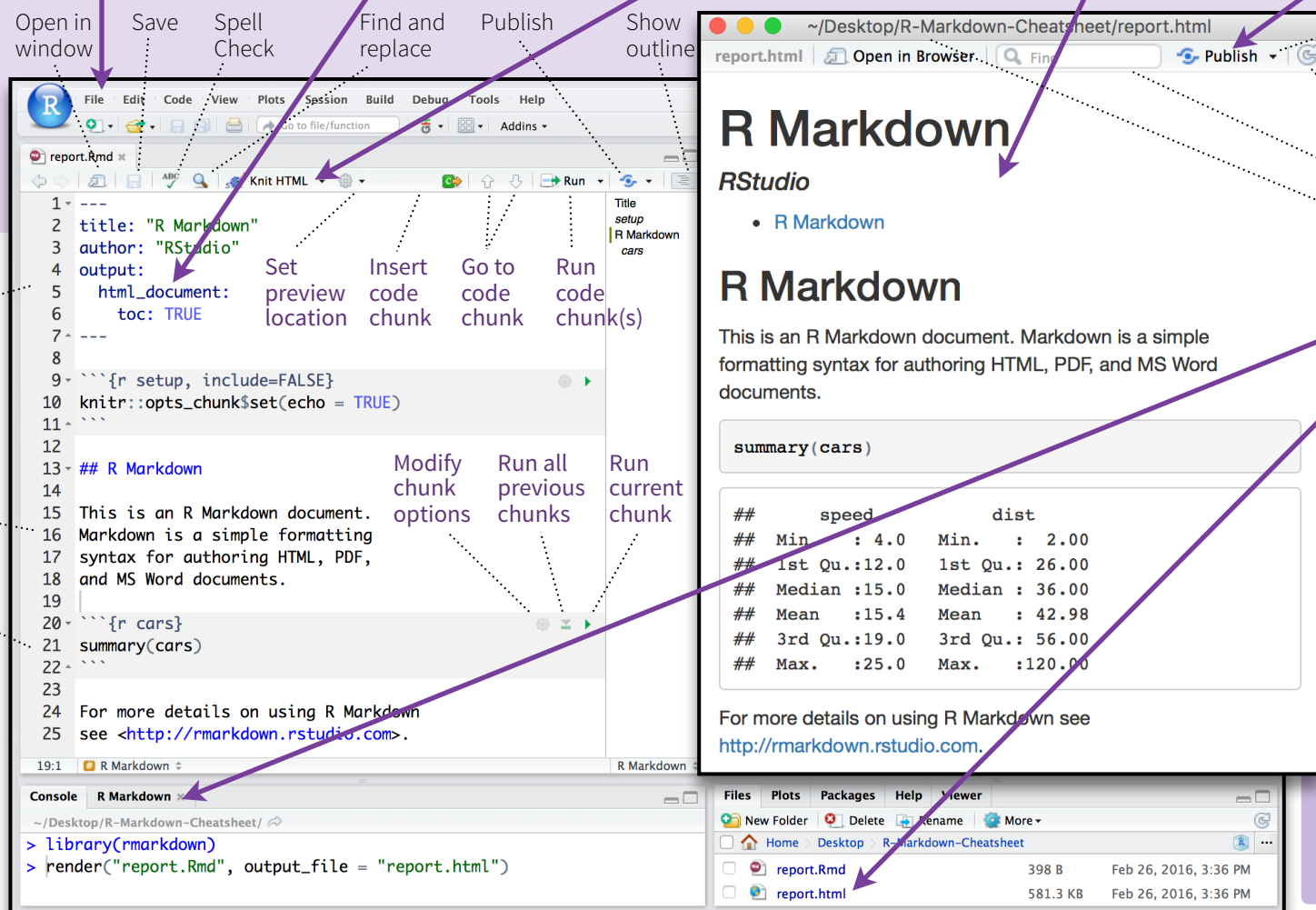
### Code chunks

Chunks of embedded code. Each chunk:

- Begins with `{r}`
- ends with `}`

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**



## 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 with `rmarkdown::run` or click **Run Document** in RStudio IDE

```
---
output: html_document
runtime: shiny
---

{r, echo = FALSE}
numericInput("n",
  "How many cars?", 5)

renderTable({
  head(cars, input$n)
})
```

How many cars?		
	5	
	speed	dist
1	4.00	2.00
2	4.00	10.00
3	7.00	4.00
4	7.00	22.00
5	8.00	16.00

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

\* Your report will be rendered as a Shiny app, which means you must choose an html output format, like **html\_document**, and serve it with an active R Session.

## Embed code with knitr syntax

### Inline code

Insert with `{r<code>}`. Results appear as text without code.

Built with  
`{r getRversion() }`

Built with 3.2.3

### Code chunks

One or more lines surrounded with `{r}` and `}`. Place chunk options within curly braces, after `r`. Insert with

```
{r echo=TRUE}
getRversion()
```

```
getRversion()
## [1] '3.2.3'
```

### Global options

Set with `knitr::opts_chunk$set()`, e.g.

```
{r include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

### 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 = 'default')

**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' - passthrough 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`, `aniopts`, `autodep`, `background`, `cache.comments`, `cache.lazy`, `cache.rebuild`, `cache.vars`, `dev`, `dev.args`, `dpi`, `engine.opts`, `engine.path`, `fig.asp`, `fig.env`, `fig.ext`, `fig.keep`, `fig.lp`, `fig.path`, `fig.pos`, `fig.process`, `fig.retina`, `fig.scap`, `fig.show`, `fig.showtext`, `fig.subcap`, `interval`, `out.extra`, `out.height`, `out.width`, `prompt`, `purl`, `ref.label`, `render`, `size`, `split`, `tidy.opts`

## Parameters

Parameterize your documents to reuse with different inputs (e.g., data sets, values, etc.)

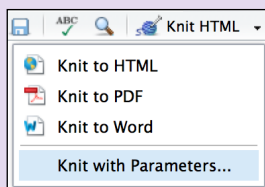
- 1 **Add parameters**  
Create and set parameters in the header as sub-values of **params**

```
---
params:
  n: 100
  d: !r Sys.Date()
---
```

- 2 **Call parameters**  
Call parameter values in code as **params\$<name>**

```
Today's date
is {r params$d}
```

- 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")))
```