

ECON 172 - Section 2 : Introduction to R

Dennis Egger and Maddie Duhon

Fall 2019

1 Welcome to R

R is a programming language and environment for statistical computing and graphics which is free to use and runs on Windows, Unix and MacOS. To do analysis in R, one can either write one's own commands or take advantage of the huge number of pre-existing packages written by R users and freely available.

1.1 Getting Started

1.1.1 Using R on your local machine (optional)

R can be downloaded from <https://cran.rstudio.com/>. We also recommend downloading RStudio, a user interface which makes R easier to use. It can be downloaded from <https://www.rstudio.com/products/rstudio/download/>.

If you run R on your local computer, you will need to install any packages you want to use (just one time), then load them into R's library for the current session. After that, it's easy to access any of the commands in the package (which are also generally extensively documented in a sort of user guide for the package).

1.1.2 Using R on your UC Berkeley's Datahub (recommended)

RStudio also runs on the UC Berkeley Datahub, so you don't need to install it locally on your computer. To start RStudio on Datahub, simply go to: <https://r.datahub.berkeley.edu/>

In our sections (and for the problem sets), we are going to show you this way of using RStudio. However, anything that we show you should also work on your local machine.

1.2 R Code

When we use R, we want to write and save our commands in a script. This will let us reproduce everything we do without having to retype our commands every time we work on a project. To do this, you can write and save a file as a .R file, which typically contains only code and code comments. You can find a separate .R file containing the code used to conduct the analysis below in the folder with the material for this section.

1.3 RMarkdown

What's even nicer about R is a file format called RMarkdown, such as the one you are currently working in. This file type allows you to combine text, mathematical equations (in a format called LaTeX), code as well as dynamic output from this code, and automatically compile / output everything into a pdf document. (In RMarkdown language, this process is called "knitting" the document.)

1.4 Setting up the Environment

First, we need to set up the R environment, including installing all the required packages, loading all the required packages. This is done using R-code. The way to let RMarkdown know that the following output is a 'code chunk' is by enclosing the code chunk in triple quotations, as below. The code chunk indicator is followed by curly brackets, and `r` (indicating this code chunk is in the R language), a name for the code chunk (here "setup"), and some additional parameters. The parameter `results='hide'` and `message=FALSE` lets RMarkdown know that we will not want to include the output of or any messages generated by this code chunk in the pdf document (but the code will still run).

In your code, you can add comments, using `#` (beginning of the line) and `##` (end of the line). These comments are not run as code, but will help your later self (and others reading your code) to understand what you were doing.

```
knitr::opts_chunk$set(echo = TRUE) ## sets global option to include code chunks in-line
#install.packages("haven") ## only run once to install
library(haven) #this library allows you to load datasets in Stata format
#install.packages("tidyverse") ## only run once to install
library(tidyverse)
#install.packages("summarytools") ## only run once to install
library(summarytools)
#install.packages("stargazer") ## only run once to install
library(stargazer) ##This package is great for making tables in .html, .tex and many other formats.
#install.packages("broom") ## only run once to install
library(broom)
```

The code chunk specifies that, by default, code chunks are part of the pdf output. Next, it installs a few packages. You only need to run this installation once. Afterwards, the package will remain installed on your datahub. (Remove the `#` before the `install.packages` commands to install each package for the first time, then “comment out” these lines using `#`, as above.) Next, the “library” command lets R know that you will want to use these packages in your code.

1.5 Loading Data

The first step in using R for statistical analysis is to store data in what R calls a “data frame,” a matrix whose columns have different modes (like numeric, words, etc.) and each row is a unit of observation (i.e: a person or a country, etc.)

R can load data from many types of files, including `.csv` and `.txt`. R has its own type of data file, with a `.rds` or `.Rdata` extension, and can also load data from Stata, SPSS, and SAS (other statistics computing programs) with the appropriate commands. The `.rds` format allows for a single object to be saved at a time, like a single dataset. The `.RData` format allows for multiple objects to be saved at once. Either one is a great way to save your data in R format.

For this section, we will be using data from Robinson, Acemoglu and Johnson (2001). For your convenience, the following link imports this data directly into your Datahub repository: https://r.datahub.berkeley.edu/hub/user-redirect/git-pull?repo=https%3A%2F%2Fgithub.com%2Fdennistegger%2FECON172_Fall2019_SectionMaterial&urlpath=rstudio%2F The folder should now appear as “ECON172_Fall2019_SectionMaterial” on your workspace (bottom right).

When we load a dataframe in R, we give it a name for reference, such as “mydata” or something more specific, like “colonials”. We also need to tell R where to find the data. We can do that by specifying a full file path to its exact location of the data. For the purposes of this section, we’ll use the dataset “colonials” which is an example dataset that contains data on property rights index, gdp per capita and other variables for countries that were European colonies in the past.

```
## Load in data in csv form, from the folder just loaded into your directory.
## Important: Paths are always relative to the RMarkdown file location
##           Alternatively, specify your working directory using
##           knitr::opts_chunk$set(root.dir = "xxx")
colonials <- read.csv("colonials.csv")
```

Next we can visualize the database we just imported, and take a look at what variables are included. To do so, it’s easiest to click directly on the dataset in your Environment pane on the top right in RStudio.

```
## Let's take a look on the dataset.
colonials ## or print first 10 observations
## Let's take a look the variables in this dataset.
```

```
names(colonials) ## just print
colonials_variables <- names(colonials) ## store list, assigned "colonials_variables"
colonials_variables ## take a look at this list
```

2 Analysis in R

2.1 Summary Statistics

Once a dataframe is loaded in R, it's very easy to run basic summary statistics. Unlike many other statistics programs, R provides fairly minimal output, and it's possible to store the results of a command without ever displaying them. For example, we may be interested in summary statistics for the GDP per capita of countries in the sample, which we can get by typing `summary(colonials$gdppc)`. This command instructs R to look in the dataframe listed before the `$` symbol (in this case, `colonials`), and to summarize the variable that follows the `$` symbol (in this case, `gdppc`). Note, the “results = ‘markup’” bit tells RMarkdown to display the output of the code exactly as displayed in the R console.

```
## Let's look at some summary statistics.
## Here is the min, 25%-ile, median, mean, 75%-ile, and max for the variable gdppc
summary(colonials$gdppc) ## just print
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      450    1480    2835    5445    6968    27330
```

```
colonials_sumstats <- summary(colonials$gdppc) ## store summary, "colonials_variables"
colonials_sumstats ## take a look at this summary
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      450    1480    2835    5445    6968    27330
```

Or, we may be interested specifically in the **mean** value of GDP per capita for countries in the sample specifically, which we can get by typing `mean(colonials$gdppc)`. As before, this command instructs R to look in the `colonials` dataframe and calculate the mean for the variable `gdppc`. We can store value resulting from this calculation, by using the `<-` symbol to store the results in a new object called `meangdp`, which can be by typing `meangdp` into the console, or used in another command or context later on.

```
## Let's look specifically at mean GDP
mean(colonials$gdppc) ## just print
```

```
## [1] 5445.458
```

```
meangdp <- mean(colonials$gdppc) ## save as a variable, called "meangdp"
meangdp ## take a look at the value stored in of "meangdp"
```

```
## [1] 5445.458
```

Similar commands exist for standard deviation (**sd**), variance (**var**), minimum (**min**), maximum (**max**), median (**med**), range (**range**) and quantile (**quantile**).

2.2 Regression

Linear regression is also very straightforward to apply in R. The `lm` command, which comes preloaded, can be used for univariate or multivariate regression:

$$Y_i = \alpha + \beta X_{1,i} + \gamma X_{2,i} + \dots + e_i$$

(Note: The above mathematical expression was written in a typesetting environment called LaTeX, and can easily be included in RMarkdown. For examples, see the Script.) Examples of how to run a regression of log GDP per capita on protection against expropriation and latitude are below:

```
## Univariate regression of log gdp per capita on property rights index.
lm(logGDP ~ protection, data=colonials) # display regression results

##
## Call:
## lm(formula = logGDP ~ protection, data = colonials)
##
## Coefficients:
## (Intercept)    protection
##      4.6604         0.5221

## Multivariate, adding absolute latitude.
lm(logGDP ~ protection + lat_abst, data=colonials) # display regression results

##
## Call:
## lm(formula = logGDP ~ protection + lat_abst, data = colonials)
##
## Coefficients:
## (Intercept)    protection    lat_abst
##      4.7281         0.4679         1.5769

## Storing results
reg1 <- lm(logGDP ~ protection, data=colonials) # display regression results
reg2 <- lm(logGDP ~ protection + lat_abst, data=colonials) # display regression results
```

In the last example, we store the results in the object reg1. If you try this in R, you'll notice the results don't display on the screen when you store the results, but you can access them by naming the object.

```
## Now access regression results by:
summary(reg1) # access complete regression results

##
## Call:
## lm(formula = logGDP ~ protection, data = colonials)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.8715 -0.4644  0.1683  0.4610  1.1413
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.66038    0.40851  11.408  < 2e-16 ***
## protection   0.52211    0.06119   8.533 4.72e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7132 on 62 degrees of freedom
## Multiple R-squared:  0.5401, Adjusted R-squared:  0.5327
## F-statistic: 72.82 on 1 and 62 DF,  p-value: 4.724e-12
```

2.3 Tables

There are many packages in R that let one create very nice tables. We recommend stargazer. Take the last regression we ran, stored as reg1 and say we want to export that to a table to be used in word or latex.

To include the final table in the knitted pdf created by RMarkdown, you can use the following template.

(Note, we want to use results='asis' and header=FALSE, in order for RMarkdown to compile the LaTeX table into a pretty format when knitting the pdf):

```
stargazer(reg1, reg2,
  out="Table 1",type="latex",header=FALSE, table.placement = "!h",
  title="Property Rights and Development",align=TRUE,
  report = "vc*st", omit.stat=c("LL","ser","f","rsq","adj.rsq"),no.space=TRUE)
```

Table 1: Property Rights and Development

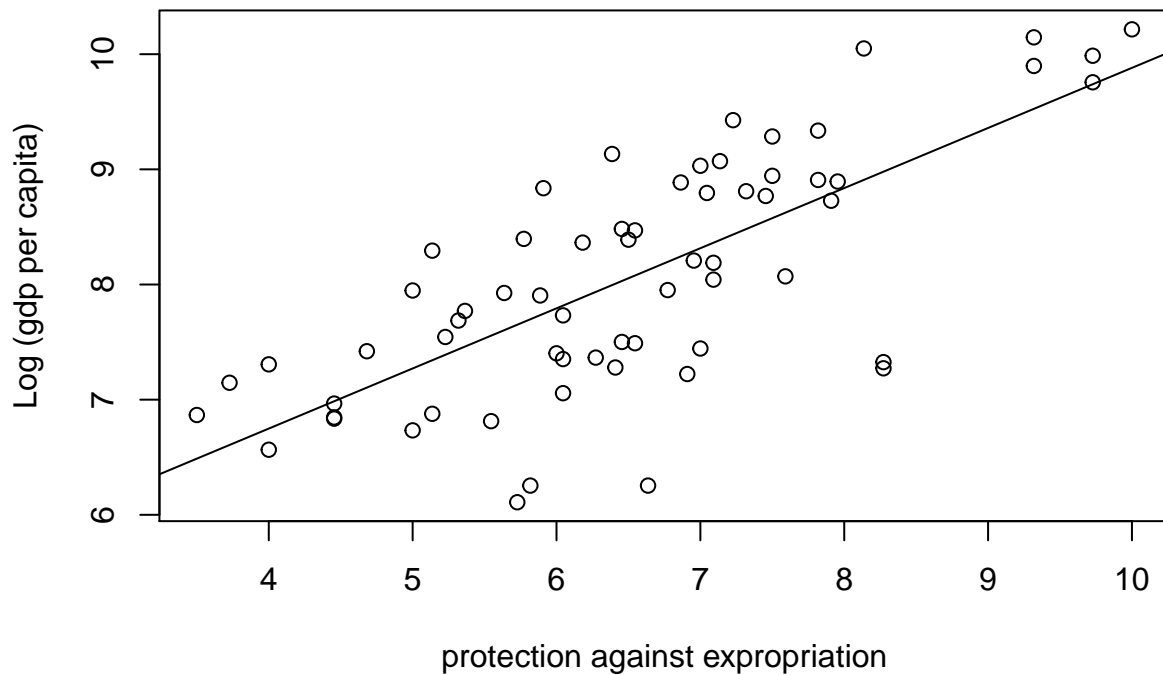
	<i>Dependent variable:</i>	
	logGDP	
	(1)	(2)
protection	0.522*** (0.061) $t = 8.533$	0.468*** (0.064) $t = 7.292$
lat_abst		1.577** (0.710) $t = 2.220$
Constant	4.660*** (0.409) $t = 11.408$	4.728*** (0.397) $t = 11.900$
Observations	64	64
<i>Note:</i> *p<0.1; **p<0.05; ***p<0.01		

The above command combines regression output saved in reg1 and reg2 into one table. Stargazer lets you specify title, alignment of coefficients inside columns and rows, which statistics to report (here, the variable names “v”, coefficients “c”, significance stars “*”, standard errors “s” and t-statistics “t”). Moreover, you can tell it to omit statistics such as the log-likelihood “LL”, the R2 “rsq”, etc. You can also set the table placement in the final pdf (here, table.placement=“H” means that the table will be placed directly after the code chunk). Stargazer has extensive documentation (try ??stargazer in R) and can also be used to make great summary statistics tables - try it out!

2.4 Plotting

R also makes it easy to do basic plots. For example, one might want to create a scatterplot from two variables, and graph the regression line - or line of best fit - on the plot. This is easy to do with the plot command, which as a default plots the first variable listed on the horizontal axis, and the second on the vertical axis. Below are two ways of plotting protection against expropriation index and log of GDP per capita, one very simple and one with titles and labels. The second version also includes the abline command to add a regression line to the graph.

```
## A scatterplot with titles
reg2 <- lm(logGDP ~ protection , data=colonials)
plot(colonials$protection,colonials$logGDP,
  xlab="protection against expropriation", ylab="Log (gdp per capita)")
abline(lm(logGDP ~ protection , data=colonials)) # line of best fit
```



Remember that abline adds to the plot the regression line (linear fit line)

3 Resources for further study

- UCLA has a number of excellent resources to help you learn more about how to use R, available at: <http://www.ats.ucla.edu/stat/r/> (<http://www.ats.ucla.edu/stat/r/>).
- You can also find interactive lessons at Try R <http://tryr.codeschool.com/> (<http://tryr.codeschool.com/>) for practice writing code.
- We also recommend the text on the syllabus, Hanck, Christoph, Martin Arnold, Alexander Gerber and Martin Schmelzer. (2018). Introduction to Econometrics with R, <https://www.econometrics-with-r.org/> (<https://www.econometrics-with-r.org/>).
- You may also find <https://www.r-bloggers.com/> (<https://www.r-bloggers.com/>) to be a useful resource for specific questions.
- Finally, we include two useful resources that summarize useful R commands and common RMarkdown features.

Types

Converting between common data types in R. Can always go from a higher value in the table to a lower value.

as.logical	TRUE, FALSE, TRUE	Boolean values (TRUE or FALSE).
as.numeric	1, 0, 1	Integers or floating point numbers.
as.character	'1', '0', '1'	Character strings. Generally preferred to factors.
as.factor	'1', '0', '1', levels: '1', '0'	Character strings with preset levels. Needed for some statistical models.

Maths Functions

log(x)	Natural log.	sum(x)	Sum.
exp(x)	Exponential.	mean(x)	Mean.
max(x)	Largest element.	median(x)	Median.
min(x)	Smallest element.	quantile(x)	Percentage quantiles.
round(x, n)	Round to n decimal places.	rank(x)	Rank of elements.
signif(x, n)	Round to n significant figures.	var(x)	The variance.
cor(x, y)	Correlation.	sd(x)	The standard deviation.

Variable Assignment

```
> a <- 'apple'
> a
[1] 'apple'
```

The Environment

ls()	List all variables in the environment.
rm(x)	Remove x from the environment.
rm(list = ls())	Remove all variables from the environment.

You can use the **environment** panel in **RStudio** to browse variables in your environment.

RStudio® is a trademark of RStudio, Inc. • [CC BY](#) Mhairi McNeill • mhairi@mcneill@gmail.com • 844-448-1212 • [rstudio.com](#)

Matrixes

```
m <- matrix(x, nrow = 3, ncol = 3)
# Create a matrix from x.
```

	m[2,] - Select a row	t(m) Transpose 
	m[, 1] - Select a column	m %*% n Matrix Multiplication 
	m[2, 3] - Select an element	solve(m, n) Find x in: m * x = n

Lists

```
l <- list(x = 1:5, y = c('a', 'b'))
# A list is collection of elements which can be of different types.
```

l[[2]]	l[1]	l\$x	l['y']
Second element of l.	New list with only the first element.	Element named x.	New list with only element named y.

Data Frames

Also see the **dplyr** library.

```
df <- data.frame(x = 1:3, y = c('a', 'b', 'c'))
# A special case of a list where all elements are the same length.
```




x	y
1	a
2	b
3	c

df\$x	df[[2]]
1 2 3	a b c

List subsetting

View(df)	head(df)
See the full data frame.	See the first 6 rows.

Matrix subsetting

df[, 2]	df[2,]	df[2, 2]
		
Number of rows.	Number of columns.	Number of columns and rows.

nrow(df)

ncol(df)

dim(df)

cbind - Bind columns.

rbind - Bind rows.

Strings

Also see the **stringr** library.

paste(x, y, sep = ' ')	Join multiple vectors together.
paste(x, collapse = ' ')	Join elements of a vector together.
grep(pattern, x)	Find regular expression matches in x.
gsub(pattern, replace, x)	Replace matches in x with a string.
toupper(x)	Convert to uppercase.
tolower(x)	Convert to lowercase.
nchar(x)	Number of characters in a string.

Factors

factor(x)
Turn a vector into a factor. Can set the levels of the factor and the order.

cut(x, breaks = 4)
Turn a numeric vector into a factor but 'cutting' into sections.

Statistics

lm(x ~ y, data=df)	t.test(x, y)	prop.test
Linear model.	Perform a t-test for difference between means.	Test for a difference between proportions.
glm(x ~ y, data=df)	pairwise.t.test	aov
Generalised linear model.	Perform a t-test for paired data.	Analysis of variance.
summary		
Get more detailed information out a model.		

Distributions

	Random Variates	Density Function	Cumulative Distribution	Quantile
Normal	rnorm	dnorm	pnorm	qnorm
Poisson	rpois	dpois	ppois	qpois
Binomial	rbinom	dbinom	pbinom	qbinom
Uniform	runif	dunif	punif	qunif

Plotting

Also see the **ggplot2** library.

	plot(x)	plot(x, y)	hist(x)
Values of x in order.	Values of x against y.	Values of x against y.	Histogram of x.

Dates

See the **lubridate** library.

Learn more at [web page](#) or [vignette](#) • package version • Updated: 3/15

R Markdown Cheat Sheet

learn more at rmarkdown.rstudio.com



Workflow

- Open a new Rmd file** at File > New File > R Markdown. Use the wizard that opens to pre-populate the file with a template.
- Write document** by editing template.
- Knit document to create report** (to knit in IDE window). Use knit button or `render()` to knit.
- Preview Output** in IDE window.
- Publish** (optional) to web or server. Synch publish button to accounts at:
 - idpubs.com
 - shinyapps.io
 - RStudio Connect
- Examine build log** in R Markdown console.
- Use output file** that is saved alongside Rmd documents.

Interactive Documents

Turn your report into an interactive Shiny document in 4 steps

- Add runtime: shiny to the YAML header.**
- Call Shiny input functions to embed input objects.**
- Call Shiny render functions to embed reactive output.**
- Render with rmarkdown::run or click Run Document in RStudio IDE**


Parameters

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

- Add parameters**
Create and set parameters in the header as sub-values of `params`.
Example: `params: n: 100 d: 1 Sys.Date()`
- Call parameters**
Call parameter values in code as `params$n` or `params$d`.
Example: `Today's date is 'r params$d'`
- Set parameters with Knit**
Set values with Knit parameters or the `params` argument of `render()`.
Example: `render("doc.Rmd", params = list(n = 1, d = as.Date("2015-01-01")))`

Learn more at: rmarkdown.rstudio.com • RStudio IDE 0.99.879 • Updated: 02/16

RStudio® is a trademark of RStudio, Inc. • 844-448-1212 • www.rstudio.com • info@rstudio.com • <http://www.rstudio.com/resources/cheatsheets/>

Pandoc's Markdown	
Write with syntax on the left to create effect on right (after render)	
Plain text	Plain text
End a line with two spaces to start a new paragraph.	End a line with two spaces to start a new paragraph.
italics and bold	<i>italics</i> and bold
verbatim code	<code>verbatim code</code>
subscript ²	sub ² supercript ²
strikethrough	--strikethrough--
escaped: * _ \	escaped: * _ \
endash: --, emdash: ---	endash: --, emdash: ---
equation: SA = pi*r^2\$	equation: SA = pi*r^2\$
equation block:	equation block:
\$E = mc^2\$	$E = mc^2$
> block quote	> block quote
## Header1 {#anchor}	Header1
### Header 2 {#css_id}	Header 2
##### Header 4	Header 3
##### Header 5	Header 4
##### Header 6	Header 5
[textbf{Text ignored in HTML}]	Header 6
HTML ignored in pdfs	HTML ignored in pdfs
<<http://www.rstudio.com>	http://www.rstudio.com
[Link](<url>, <title>)	Jump to Header 1
Jump to [Header 1]{#anchor}	link
:image:	Image:
!<Caption>(smallorb.png)	
* unordered list	Caption
+ sub-item 1	• unordered list
+ sub-item 2	• sub-item 1
- sub-sub-item 1	• sub-item 2
* item 2	• item 2
Continued (indent 4 spaces)	Continued (indent 4 spaces)
1. ordered list	1. ordered list
2. item 2	2. item 2
A. sub-item 1	1. sub-item 1
A. sub-sub-item 1	A. sub-sub-item 1
(@) A list whose numbering continues after	continues after
(@) an interruption	2. an interruption
Term 1	Term 1
:	Definition 1
: Definition 1	Right Left Default Center
Right Left Default Center	12 12 12 12
12 12 12 12	123 123 123 123
1 1 1 1	1 1 1 1
- slide bullet 1	• side bullet 1
- slide bullet 2	• side bullet 2
(>) - to have bullets appear on click	(>) - to have bullets appear on click
horizontal rule/slide break:	horizontal rule/slide break:

A footnote [1]	A footnote 1
[1]: Here is the footnote.	1. Here is the footnote.

Set render options with YAMl

When you render R Markdown

1. runs the R code, embeds results and text into .md file with knitr
2. then converts the .md file into the finished format with pandoc

```
---
output: html_document
# Body
---
```

creates

html_document
pdf (requires Tex)
Microsoft Word (docx)
OpenDocument Text
Rich Text Format
Markdown
Github compatible markdown
ioslides HTML slides
slidy HTML slides
beamer presentation (requires Tex)

output value

html_document
pdf_document
word_document
odt_document
rtf_document
md_document
github_document
ioslides_presentation
slidy_presentation
beamer_presentation

Customize output with sub-options (listed at right):

```
---
indent2: spaces
indent4: spaces
output:
  html_document:
    code_folding: hide
    toc_float: TRUE
---
# Body
```

html tabs

Use tabset css class to place sub-headers into tabs

```
## Tabset {.tabset .tabset-fade .tabset-pills}
## Tab 1
text 1
## Tab 2
text 2
## End tabset
```

sub-option	description
citation_package	The LaTeX package to process citations, natbib, biblatex or none
code_folding	Let readers to toggle the display of R code, "none", "hide", or "show"
colortheme	Beamer color theme to use
css	CSS file to use to style document
dev	Graphics device to use for figure output (e.g. "png")
duration	Add a countdown timer (in minutes) to footer of slides
fig_caption	Should figures be rendered with captions?
fig_height	Default figure height and width (in inches) for document
fig_width	Syntax highlighting: "tango", "pygments", "kate", "zenburn", "textmate"
highlight	File of content to place in document (in .header, .before_body, .after_body)
includes	Should bullets appear one at a time (on presenter mouse clicks)?
incremental	Save a copy of .md file that contains knitr output
keep_md	Save a copy of text file that contains knitr output
keep_tex	Engine to render latex, "pdf latex", "xelatex", or "lua latex"
latex_engine	Directory of dependency files to use (Bootstrap, MathJax, etc.)
lib_dir	Set to Local or a URL to use a local/URL version of MathJax to render
mathjax	Markdown extensions to add to default definition of R Markdown
md_extensions	Add section numbering to headers
number_sections	Additional arguments to pass to Pandoc
pandoc_args	Preserve YAML front matter in final document?
preserve_yaml	doc file whose styles should be copied when producing docx output
reference_docx	Embed dependencies into the doc
self_contained	The lowest heading level that defines individual slides
slide_level	Use the smallest font size in the presentation?
smaller	Convert straight quotes to curly, dashes to em-dashes, ... to ellipses, etc.
smart	Pandoc template to use when rendering file
template	Bookdown or Beamer theme to use for page
theme	Add a table of contents at start of document
toc	The lowest level of headings to add to table of contents
toc_depth	Float the table of contents to the left of the main content
toc_float	Options not listed: extra_dependencies, fig_crop, fig_retina, font_adjustment, font_theme, footer, logo, html_preview, reference_odt, transition, variant, widescreen

Set render options with YAMl

When you render R Markdown

1. runs the R code, embeds results and text into .md file with knitr
2. then converts the .md file into the finished format with pandoc

```
---
output: html_document
# Body
---
```

creates

html_document
pdf (requires Tex)
Microsoft Word (docx)
OpenDocument Text
Rich Text Format
Markdown
Github compatible markdown
ioslides HTML slides
slidy HTML slides
beamer presentation (requires Tex)

output value

html_document
pdf_document
word_document
odt_document
rtf_document
md_document
github_document
ioslides_presentation
slidy_presentation
beamer_presentation

Customize output with sub-options (listed at right):

```
---
indent2: spaces
indent4: spaces
output:
  html_document:
    code_folding: hide
    toc_float: TRUE
---
# Body
```

html tabs

Use tabset css class to place sub-headers into tabs

```
## Tabset {.tabset .tabset-fade .tabset-pills}
## Tab 1
text 1
## Tab 2
text 2
## End tabset
```

sub-option	description
citation_package	The LaTeX package to process citations, natbib, biblatex or none
code_folding	Let readers to toggle the display of R code, "none", "hide", or "show"
colortheme	Beamer color theme to use
css	CSS file to use to style document
dev	Graphics device to use for figure output (e.g. "png")
duration	Add a countdown timer (in minutes) to footer of slides
fig_caption	Should figures be rendered with captions?
fig_height	Default figure height and width (in inches) for document
fig_width	Syntax highlighting: "tango", "pygments", "kate", "zenburn", "textmate"
highlight	File of content to place in document (in .header, .before_body, .after_body)
includes	Should bullets appear one at a time (on presenter mouse clicks)?
incremental	Save a copy of .md file that contains knitr output
keep_md	Save a copy of text file that contains knitr output
keep_tex	Engine to render latex, "pdf latex", "xelatex", or "lua latex"
latex_engine	Directory of dependency files to use (Bootstrap, MathJax, etc.)
lib_dir	Set to Local or a URL to use a local/URL version of MathJax to render
mathjax	Markdown extensions to add to default definition of R Markdown
md_extensions	Add section numbering to headers
number_sections	Additional arguments to pass to Pandoc
pandoc_args	Preserve YAML front matter in final document?
preserve_yaml	doc file whose styles should be copied when producing docx output
reference_docx	Embed dependencies into the doc
self_contained	The lowest heading level that defines individual slides
slide_level	Use the smallest font size in the presentation?
smaller	Convert straight quotes to curly, dashes to em-dashes, ... to ellipses, etc.
smart	Pandoc template to use when rendering file
template	Bookdown or Beamer theme to use for page
theme	Add a table of contents at start of document
toc	The lowest level of headings to add to table of contents
toc_depth	Float the table of contents to the left of the main content
toc_float	Options not listed: extra_dependencies, fig_crop, fig_retina, font_adjustment, font_theme, footer, logo, html_preview, reference_odt, transition, variant, widescreen