# Reporting Using R and Markdown:

# Part II: knitr

Niels Hagenbuch

Roche and Constat.ch

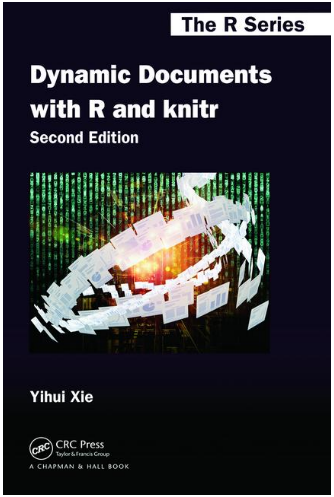
26 August 2025

## knitr

1. Books
2. Idea
3. Control Sequences
4. R Chunks
5. Tables
6. Plots
7. Line Numbers for Code Blocks
8. Commenting-out
9. Caching
10. R Code in Chunk Options

## Books

## Books



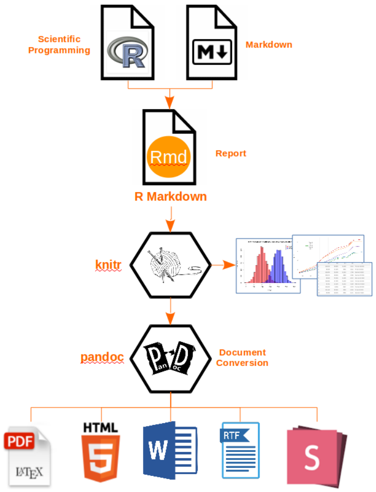
Yihui Xie
**Dynamic Documents with R and knitr**
2nd Edition
CRC Press, 2015

ISBN 978-1-4987-1696-3

Summaries of **Options** and **Hooks** are available on the Web:
https://yihui.org/knitr/

## Idea

## Idea



See also:

https://stackoverflow.com/questions/40563479/relationship-between-r-markdown-knitr-pandoc-and-bookdown

## Concept

- A **kntr file** (`.Rmd`) contains **R Markdown code** and **R code**.
- The **Markdown part** contains your
  - text
  - tables
  - images
  - footnotes
- The **R parts** contain the code for
  - handling the data
  - creating content for tables
  - creating plots
  - statistical analyses

## Concept

Uniting these two parts is called **“to knit.”**

This is a process run in R, the package *knitr* is required (comes by default with RStudio; outside RStudio it needs to be installed).

1. In a **first pass**, R code in the `.Rmd` file is executed.
2. Control sequences to mark R code are removed, and the R code is replaced by its result:
   - a number
   - a table
   - a plot
   - a summary
   - nothing
3. In a **second pass**, the R Markdown text, now containing the results of R as text, is converted to HTML, Word, or PDF.

## Concept

Extracting only the R code from an `.Rmd` file is called **"to purl."**

1. Any R Markdown text is removed.
2. The remaining R code from chunks is properly formatted and commented.
3. The ensuing `.R` file contains only the R part of your report and can be run separately.

**Note:** All embedded in-line R code is lost!

## Markdown

→ Open file `knitrExample.Rmd`

## Control Sequences

## Control Sequences

knitr goes through an `.Rmd` document line by line.

It ignores text and Markdown code, but **recognises and executes R code**, if it is enclosed within two **control sequences**.

There are two ways to include R code:

- In-line R code
- R chunks

## Control Sequences: In-line R Code

- **In-line R code** is enclosed in an opening `` `r `` and closing backtick `` ` ``
- The in-line “mini-chunk” must not contain line-breaks.
- Should your editor introduce line-breaks, remove any space in the R code or put it on a separate line.

```
The value of x is `r round(x <- sqrt(2), 4)` and displayed here.

After the transformation, x is
`r print(exp(asin(tan(cos(sin((x))))))-128/365.25*x^3)`.
```

## Control Sequences: In-line R Code

```
The value of x is `r round(x <- sqrt(2), 4)` and
displayed here.

After the transformation, x is
`r print(exp(asin(tan(cos(sin((x))))))-128/365.25*x^3)`.
```

becomes...

```
The value of x is 1.4142 and displayed here.

After the transformation, x is 0.9454325.
```

## Control Sequences: R Chunks

- An **R chunk** begins with ` ```{r } ` and ends with ` ``` `
- Both control sequences have to be at the beginning of a separate line.
- The opening ` ```{r ...} ` must not contain line-breaks.
- In-between, any number of lines of R code can be stated.

~~~
```{r chunk name, chunk options}
x <- sqrt(2)
round(x, 4)

x.tr <- exp(asin(tan(cos(sin((x)))))) - 128/365.25 * x^3
x.tr
```
~~~

## R Chunks

## knitr: Chunk Syntax

knitr offers many options to control the way R chunks are processed and how results are transformed into an R Markdown text.

The **general syntax** is:

~~~
```{r chunkname, chunk options}
# Your
# lines
# of
# R code
```
~~~

## knitr: Chunk Names

~~~
```{r chunkname, chunk options}
# some R code
```
~~~

**Chunk Names**

- Optional, although highly recommended (for debugging)
- Must be **unique**
- No spaces
- Hyphen - and underscore _ are allowed
- Numbers are allowed (even at the beginning of the name!)
- (No periods)

## knitr: Chunk Options

All **Chunk Options** are explained here:

https://yihui.org/knitr/options/

The most important are:

| **Option** | **Description** |
|---|---|
| `eval` | Evaluate/run the chunk of R code? |
| `echo` | Display the *R code* in the document? |
| `include` | Display the *result* of the R code? |
| `message` | Show messages in the document? |
| `warning` | Show warning messages in the document? |
| `error` | Show error messages in the document? |

- All these options expect a logical as argument (e.g., `echo = FALSE`).
- The default for the options listed above is `TRUE`.

## In the Beginning. . .

When starting an analysis that is likely to turn into a report, start with an .Rmd file including title, author, and date.

Then develop your code within a chunk that is neither evaluated nor shown.

RStudio switches to “R mode” when the cursor is within an R chunk.

~~~
```{r SANDBOX, eval = FALSE, echo = FALSE}
# Develop your
# R code
# here
```
~~~

## Set General Options

The two options `echo` and `include` are usually set to `FALSE` to avoid displaying R code and unnecessary results.

To set knitr options for the entire document, use the function `opts_chunk$Set()`

An `.Rmd` file may begin with the following two R chunks:

~~~
```{r knitr-setup, include = FALSE}
library(knitr)
opts_chunk$set(echo = FALSE, include = FALSE)
```
~~~

~~~
```{r R-setup}
options(show.signif.stars = FALSE)
rnd <- 2   ##  Number of decimals to round.
```
~~~

## Example

R code and output to be displayed must be “turned on manually”:

The following two linear models were fitted in R:

~~~
```{r Regressions, echo = TRUE}
m1 <- lm(No.of.Births ~ 1 + No.of.Birds, data = Baby)
m2 <- lm(No.of.Births ~ 1 + No.of.Birds + Year, data = Baby)
```
~~~

The unadjusted association of number of birds and birth rate is:

~~~
```{r RegrModel1_Summary, include = TRUE}
summary(m1)
```
~~~

## Tables

## Tables

Creating tables is a bit of a challenge.

A comprehensive package is sjPlot:
`https://strengejacke.github.io/sjPlot/index.html`

Smaller functions to create tables:

| Package | Function | Description |
|---|---|---|
| knitr | `kable` | Simple tables in Word, HTML, and LaTeX |
| kableExtra | `kable_styling` | Enhance `kable` with different styles in HTML and LaTeX |
| flextable | | For complex tables (best for Word) |
| table1 | `table1` | “Table 1” for biomedical papers (results are in HTML or LaTeX) |

## Very Simple Tables with kable

knitr includes the function `kable()`, which produces simple tables.

~~~
```{r Kable, include = TRUE}
iris$noise <- rnorm(n = nrow(iris))
kable(iris[1:10, -5])
```
~~~

## Very Simple Tables with kable

Useful parameters:

- `digits` controls the number of decimals
- `align` controls the alignment of the columns (string of `l`, `c`, and `r`)
- `caption` prints a caption *above* the table

~~~
```{r Kable2, include = TRUE}
kable(iris[1:10, -5], digits = 4,
      align = c("llccr"),
      caption = "Tab. 1: First 10 Observations
                 in the Famous Iris Data Set.")
```
~~~

## Plots

## Plots

To include plots in the document, the chunk options need to be set to `include = TRUE`

The plot is directly included in the HTML/Word/PDF document.

~~~
```{r Sine, include = TRUE}
plot(sin, -pi, 2 * pi)
title(main = substitute(paste("Sine from ", -pi, " to 2", pi)))
```
~~~

## Plots: Files

A better way is to use the options fig.path (where to store the file) and dev (file format; see https://yihui.org/knitr/options/#plots) in the initial knitr-setup chunk.

~~~
```{r knitr-setup2, include = FALSE}
opts_chunk$set(echo = FALSE, include = FALSE,
               fig.path = "Figures/", dev = "jpeg")
```
~~~

The **chunk name** will be used as **file name**, and the plot is available in the directory indicated by fig.path.

~~~
```{r Sine2, include = TRUE}
plot(sin, -pi, 2 * pi)
title(main = substitute(paste("Sine from ", -pi, " to 2", pi)))
```
~~~

## Plots: Size

The size (in inches) is controlled by the two options `fig.height` and `fig.width`.

~~~
```{r SineSmall, include = TRUE, fig.height = 3, fig.width = 3}
plot(sin, -pi, 2 * pi)
```
~~~

## Plots: Captions

A caption beneath the plot can be added using the option `fig.cap`.

~~~
```{r Cosine, include = TRUE, fig.cap = "Fig. 2: A Cosine Curve."}
plot(cos, -pi, 2 * pi)
```
~~~

## Line Numbers for Code Blocks

## Line Numbers for Code Blocks

With the chunk option `attr.source = ".numberLines"` you can add line numbers to code.

In the YAML section at the beginning of the R Markdown document, a syntax highlight theme needs to be provided (notice the mandatory indentation with 2 and 4 spaces):

```
output:
  html_document:
    highlight: tango
```

For a list of syntax highlight themes, see:
https://bookdown.org/yihui/rmarkdown/appearance-and-style-1.html

In the chunks, the options `echo` and `include` need to be set to `TRUE`, even if there is no output.

The numbers start with 1 in each code block.

## Line Numbers for Code Blocks

~~~
```{r, echo = TRUE, include = TRUE, attr.source = ".numberLines"}
##  Load the data set.
data(cars)

##  Transform the var. 'speed'.
cars$speed2 <- cars$speed^2

##  Analysis.
fm <- lm(dist ~ 1 + speed + speed2, data = cars)
#plot(fm)
summary(fm)
```
~~~

For the result, see the rendered HTML output.

## Commenting-out

## Commenting-out

**On HTML level: showing and hiding text**

- In-line R code ("mini-chunks") and R Chunks can be out-commented using HTML's `<!-- -->`
- HTML comments do not deactivate R code: knitr ignores these tags, executes the R code, and inserts potential results.
- The comment tags remain in the Markdown document and **suppress displaying the results** of R (which are included in the HTML file).

**On R level: turning code on and off**

- Use # in in-line code (e.g., `` `r #foo(bar, baz)` ``).
- Use `eval = FALSE` to deactivate entire chunks.

## Caching

## Caching

The option `cache` allows knitr to store the results of the chunk.

The next time the file is knitted, the results are obtained from the cache (files on your computer) instead of running the R code again.

`cache = TRUE`:

- If the chunk is run for the first time, the results are stored.
- If stored results exist, they will be used. No code is run.
- If the chunk is modified, it will be re-run and the results are stored. **Note:** *Any* change in the code, chunk options, or name is considered a modification (e.g., the removal of a space, adding a blank line).

`cache = FALSE`:

- R code is run; existing cached results are ignored.
- No results are cached.

## Caching: Tremendous Speed-up

Cached results can **save a lot of time!**

To knit a completely cached document takes no longer than pasting the stored R outputs into the document and running R Markdown.

~~~
```{r MultipleImputation, cache = TRUE}
imp <- mice(data.na, m = 1000, maxit = 50, ...)
fit <- with(imp, lmer(Y ~ 1 + X1 * X2 * X3 + (1 | ID)))
summary(pool(fit))
```
~~~

## Caching: Pitfall 1

Cached results can **cause subtle errors and unexpected results!**

~~~
```{r Chunk-1, cache = TRUE}
res <- foo(x)
```
~~~

A change in `Chunk-1` is **not propagated** into `Chunk-2` and `Chunk-3`:

~~~
```{r Chunk-2, cache = TRUE}
plot(res)
```
~~~

~~~
```{r Chunk-3, cache = TRUE}
summary(res)
```
~~~

## Caching: Pitfall 1

Use a global flag and set it when required.

~~~
```{r knitr-setup, include = FALSE}
library(knitr)
opts_chunk$set(echo = FALSE, include = FALSE)

##  Global option for chunk caching.
cache.flag = FALSE
#cache.flag = TRUE

##  Global option for figure caching.
fig.cache = FALSE
#fig.cache = TRUE
```
~~~

## Caching: Pitfall 1

Use the global flag variables as arguments for option cache.

~~~
```{r Chunk-1, cache = cache.flag}
res <- foo(x)
```
~~~

~~~
```{r Chunk-2, cache = fig.cache}
plot(res)
```
~~~

~~~
```{r Chunk-3, cache = cache.flag}
summary(res)
```
~~~

## Caching: Pitfall 1

Use the global flag variables as arguments for option cache.

~~~
```{r Regression, cache = cache.flag}
fm <- lm(No.of.Births ~ 1 + No.of.Birds, data = Baby)
```
~~~

~~~
```{r RegrPlot, include = TRUE, cache = fig.cache}
plot(fm)
```
~~~

~~~
```{r MultipleImputation, cache = MICE.cache}
imp <- mice(data.na, m = 1000, maxit = 50, ...)
fit <- with(imp, lmer(Y ~ 1 + X1 * X2 * X3 + (1 | ID)))
summary(pool(fit))
```
~~~

## Caching: Pitfall 2

If, **after caching is turned off**, changes are made in R chunks at the beginning of the knitr file that should propagate throughout the analysis,

and caching is **turned on** again later,

the **old content** is used.

**“When in doubt, delete cache!”**

## Caching: Dependencies

The option `dependson` allows to define dependencies.

`dependson` overrules `cache`.

~~~
```{r Chunk-1, cache = cache.flag}
res <- foo(x)
```
~~~

~~~
```{r Chunk-2, cache = cache.flag, dependson = "Chunk-1"}
plot(res)
```
~~~

~~~
```{r Chunk-3, cache = cache.flag, dependson = "Chunk-1"}
summary(res)
```
~~~

**Cave: Spaghetti-code and dangling/wild references!**

## R Code in Chunk Options

## R Code in Chunk Options

Options in the chunk header have the form `tag = value`

The `value` part is parsed as R code:

- Options accept variables (see also pp. 41, 42, and 43).
- R functions can be called.

~~~
```{r PlotSettings}
fig.h <- 4; fig.w <- 6; scl <- 1   ##  Settings for HTML.
scl <- 2   ##  Settings for PDF.
```
~~~

~~~
```{r PlotO3, include = TRUE, fig.height = fig.h * scl, fig.width = fig.w * scl,
      fig.cap = paste0("The plot is based on ", nrow(data), " measurements.")}
plot(Ozone ~ Temp, data = airquality, ...)
```
~~~

For more details, see https://yihui.org/knitr/options/

# References

Xie, Yihui (2015). *Dynamic Documents with R and knitr*. Second Edition. CRC Press. ISBN: 978-1-4987-1696-3.

Xie, Yihui, J. J. Allaire, and Garrett Grolemund (2018). *R Markdown. The Definitive Guide*. CRC Press. ISBN: 978-1-138-35933-8. URL: `https://bookdown.org/yihui/rmarkdown/`.