Writing reproducible reports

knitr with R Markdown

Karl Broman

Biostatistics & Medical Informatics, UW–Madison

kbroman.org
github.com/kbroman
@kwbroman

Course web: kbroman.org/Tools4RR

knitr in a knutshell

kbroman.org/knitr_knutshell

Data analysis reports

- Figures/tables + email
- Static LaTeX or Word document
- knitr/Sweave + LaTeX $\rightarrow$ PDF
- knitr + Markdown $\rightarrow$ Web page

What if the data change?

What if you used the wrong version of the data?

knitr code chunks

Input to knitr:

```
We see that this is an intercross with `r nind(sug)`
individuals. There are `r nphe(sug)` phenotypes, and genotype
data at `r totmar(sug)` markers across the `r nchr(sug)`
autosomes.  The genotype data is quite complete.

```{r summary_plot, fig.height=8}
plot(sug)
```
```

Output from knitr:

```
We see that this is an intercross with 163
individuals. There are 6 phenotypes, and genotype
data at 93 markers across the 19
autosomes.  The genotype data is quite complete.

```r
plot(sug)
```

![plot of chunk summary_plot](RmdFigs/summary_plot.png)
```

html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset=utf-8"/>
  <title>Example html file</title>
</head>

<body>
<h1>Markdown example</h1>

<p>Use a bit of <strong>bold</strong> or <em>italics</em>. Use
backticks to indicate <code>code</code> that will be rendered
in monospace.</p>

<ul>
<li>This is part of a list</li>
<li>another item</li>
</ul>

</body>
</html>
```

[Example]

CSS

```
ul,ol {
  margin: 0 0 0 35px;
}

a {
  color: purple;
  text-decoration: none;
  background-color: transparent;
}

a:hover
{
  color: purple;
  background: #CAFFFF;
}
```

[Example]

Markdown

````
# Markdown example

Use a bit of **bold** or _italics_. Use backticks to indicate
`code` that will be rendered in monospace.

- This is part of a list
- another item

Include blocks of code using three backticks:

```
x <- rnorm(100)
```

Or indent four spaces:

    mean(x)
    sd(x)

And it's easy to create links, like to
[Markdown](http://daringfireball.net/projects/markdown/).
````

[Example | MD cheat sheet]

R Markdown

- R Markdown is a variant of Markdown, developed at RStudio.com
- Markdown + knitr + extras
- A few extra marks
- LaTeX equations
- Bundle images into the final html file

Code chunks, again

````
```{r knitr_options, include=FALSE}
knitr::opts_chunk$set(fig.width=12, fig.height=4,
                      fig.path='Figs/', warning=FALSE,
                      message=FALSE)
set.seed(53079239)
```

### Preliminaries

Load the R/qtl package using the `library` function:

```{r load_qtl}
library(qtl)
```

To get help on the read.cross function in R, type the
following:

```{r help, eval=FALSE}
?read.cross
```
````

[Example]

Chunk options

`echo=FALSE`	Don't include the code
`results="hide"`	Don't include the output
`include=FALSE`	Don't show code or output
`eval=FALSE`	Don't evaluate the code at all
`warning=FALSE`	Don't show R warnings
`message=FALSE`	Don't show R messages
`fig.width=#`	Width of figure
`fig.height=#`	Height of figure
`fig.path="Figs/"`	Path for figure files

There are lots of chunk options.

Global chunk options

````
```{r knitr_options, include=FALSE}
knitr::opts_chunk$set(fig.width=12, fig.height=4,
                      fig.path='Figs/', warning=FALSE,
                      message=FALSE, include=FALSE,
                      echo=FALSE)
set.seed(53079239)
```

```{r make_plot, fig.width=8, include=TRUE}
x <- rnorm(100)
y <- 2*x + rnorm(100)
plot(x, y)
```
````

- Use global chunk options rather than repeat the same options over and over.
- You can override the global values in specific chunks.

Package options

```
```{r package_options, include=FALSE}
knitr::opts_knit$set(progress = TRUE, verbose = TRUE)
```
```

- It's easy to confuse global chunk options with package options.
- I've not used package options.
- So focus on `opts_chunk$set()` not `opts_knit$set()`.

In-line code

```
We see that this is an intercross with `r nind(sug)`
individuals. There are `r nphe(sug)` phenotypes, and genotype
data at `r totmar(sug)` markers across the `r nchr(sug)`
autosomes.  The genotype data is quite complete.
```

- Each bit of in-line code needs to be within one line; they can't span across lines.
- I'll often precede a paragraph with a code chunk with `include=FALSE`, defining various variables, to simplify the in-line code.
- Never hard-code a result or summary statistic again!

YAML header

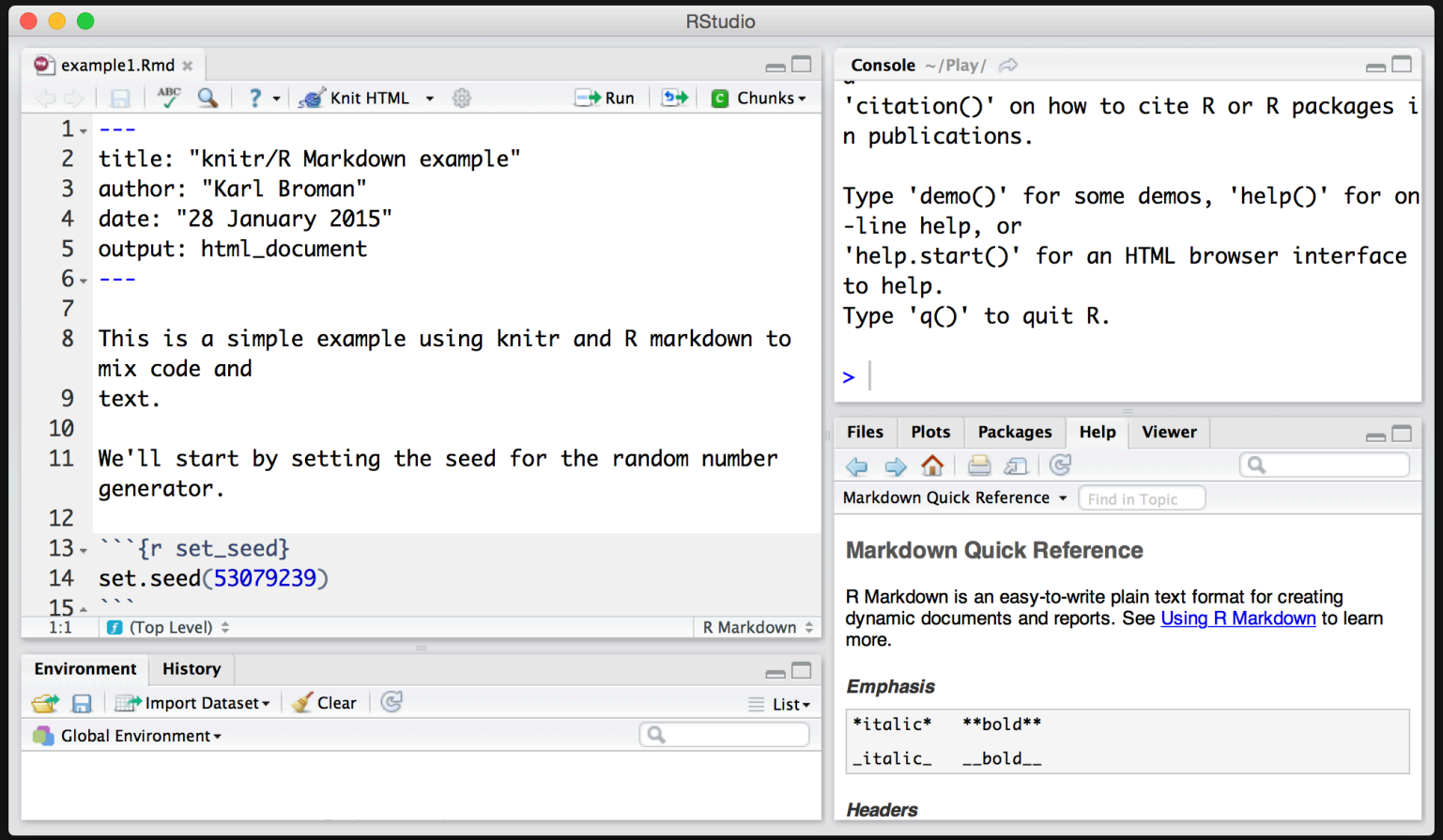
```
---
title: "knitr/R Markdown example"
author: "Karl Broman"
date: "28 January 2015"
output: html_document
---
```

```
---
title: "Another knitr/R Markdown example"
author: "[Karl Broman](http://kbroman.org)"
date: "`r Sys.Date()`"
output: word_document
---
```

Rounding

- `cor(x,y)` might produce `0.8992877`, but I want `0.90`.
- `round(cor(x,y), 2)`, would give `0.9`, but I want `0.90`.
- You could use `sprintf("%.2f", cor(x,y))`, but `sprintf("%.2f", -0.001)` gives `-0.00`.
- Use the `myround` function in my R/broman package.
- `myround(cor(x,y), 2)` solves both issues.

R Markdown → html, in RStudio



R Markdown $\rightarrow$ html, in R

```
> library(rmarkdown)
> render("knitr_example.Rmd")
```

```
> rmarkdown::render("knitr_example.Rmd")
```

R Markdown → html, GNU make

```
knitr_example.html: knitr_example.Rmd
    R -e "rmarkdown::render('knitr_example.Rmd')"
```

Need pandoc in your `PATH`

RStudio includes pandoc; you just need to add the relevant directory to your `PATH`.

Mac:

```
/Applications/RStudio.app/Contents/MacOS/pandoc
```

Windows:

```
"c:\Program Files\RStudio\bin\pandoc"
```

Reproducible knitr documents

- Don't use absolute paths like `~/Data/blah.csv`
- Keep all of the code and data in one directory (and its subdirectories)
- If you must use absolute paths, define the various directories with variables at the top of your document.
- Use `R --vanilla` or perhaps

```
R --no-save --no-restore --no-init-file --no-site-file
```

- Use GNU make to document the construction of the final product (tell future users what to do)
- Include a final chunk with `getwd()` and `devtools::session_info()`.
- For simulations, use `set.seed` in your first chunk.

Controlling figures

````
```{r test_figure, dev.args=list(pointsize=18)}
x <- rnorm(100)
y <- 2*x + rnorm(100)
plot(x,y)
```
````

- The default is for knitr/R Markdown is to use the `png()` graphics device.
- Use another graphics device with the chunk option `dev`.
- Pass arguments to the graphics device via the chunk option `dev.args`.

Tables

````
```{r kable}
x <- rnorm(100)
y <- 2*x + rnorm(100)
out <- lm(y ~ x)
coef_tab <- summary(out)$coef
library(kable)
kable(coef_tab, digits=2)
```
````

````
```{r pander}
library(pander)
panderOptions("digits", 2)
pander(out, caption="Regression coefficients")
```
````

````
```{r xtable, results="asis"}
library(xtable)
tab <- xtable(coef_tab, digits=c(0, 2, 2, 1, 3))
print(tab, type="html")
```
````

Important principles

Modify your desires to match the defaults.

Focus your compulsive behavior on things that matter.