Writing reproducible reports
knitr with R Markdown

Karl Broman

Biostatistics & Medical Informatics, UW–Madison

kbroman.org
github.com/kbroman
@kwbroman

Course web: kbroman.org/AdvData
How many simulation replicates?
How many simulation replicates?

- To estimate power?
How many simulation replicates?

- To estimate power?
- To estimate a p-value?
How many simulation replicates?

- To estimate power?
- To estimate a p-value?
- To estimate some other quantity?
Data analysis reports

- Figures/tables + email
- Static Word document
- \LaTeX + R → PDF
- R Markdown = knitr + Markdown → Web page
What if the data change?

What if you used the wrong version of the data?
knitr in a nutshell

kbroman.org/knitr_knutshell
knitr in a nutshell

kbroman.org/knitr_knutshell

rmarkdown.rstudio.com
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
```
```
```

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.

```
```
```
```
```

![plot of chunk summary_plot](RmdFigs/summary_plot.png)
<!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>
ul, ol {
  margin: 0 0 0 35px;
}

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

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

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

Or indent four spaces:

```r
tmp <- mean(x)
```

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

- toc_float
- toc_depth
- code_folding
- theme
- df_print
### Preliminaries

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

```r
library(qtl)
```

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

```r
?read.cross
```
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

- Use global chunk options rather than repeat the same options over and over.

- You can override the global values in specific chunks.
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()`.
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!
You can have python code chunks in R Markdown. And information is remembered between chunks.

```
```{python define_something}
x = [2, 3, 5, 7, 9, 11, 13, 17]
```

```
```{python list_comprehension}
y = [v*2 for v in x]
```

It seems like you can’t use python in-line. But if load the package ‘reticulate‘, you can get access to python objects with R code.

The first value in `x` is `r py$x[1]`, while the first value in `y` is `r py$y[1]`.

More at rstudio.github.io/reticulate/
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
---
title: "knitr/R Markdown example"
author: "Karl Broman"
date: "28 January 2015"
output: html_document
---

This is a simple example using knitr and R markdown to mix code and text.

We'll start by setting the seed for the random number generator.

```{r set_seed}
set.seed(53079239)
```
R Markdown → html, in R

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

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

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

```{r gt}
library(gt)
gt( round(coef_tab, 2) )
```
Modify your desires to match the defaults.

Focus your compulsive behavior on things that matter.
What should a report contain?
Karl -- this is very interesting, however you used an old version of the data (n=143 rather than n=226).

I'm really sorry you did all that work on the incomplete dataset.

Bruce
What should a report contain?

- Explain your shared goals
- Describe the data
- Explain what you did
- Show your results
- Explain your conclusions
- When you’re done, go back and write an *executive summary*
Standard scientific article

- Abstract
- Introduction/background
- Materials and methods
- Results
- Conclusions/discussion

Why this format?
Further suggestions

- Tailor the report to the audience
- Try not to be boring
- Limit equations and code; details in an appendix
- Break it up into sections; simple and clear language and structure
- Lots of figures, ideally interactive; explain the figures
- What do + and – mean (regarding coefficients/effects)?
Organizing projects

▶ RStudio Projects
▶ here package for R
Other R Markdown-based things

▶ blogdown for websites
▶ bookdown for book-like objects
▶ xaringan for slides
▶ pagedown for paged documents (like resumes or letters)
Interactive graphics tools

- plotly
- htmlwidgets
- leaflet
- networkD3
- DiagrammeR
- DT
- d3heatmap
- scatterD3