This lecture is about how to create reproducible manuscripts, for journal articles. KnitR with R Markdown is great for informal reports. KnitR with AsciiDoc is great for somewhat fancier reports. There are a number of efforts, especially with Pandoc, to use R Markdown for journal articles. But if you want fine control over the appearance of a document, it’s hard to beat \LaTeX, and so I’m just going to focus on that.

I can’t hope to explain \LaTeX properly in just this one lecture. My goals are to give the general gist, indicate resources and options, and show how to use KnitR with \LaTeX.

I also want to discuss some more general strategies for ensuring that the results described in a journal article are fully reproducible.
\documentclass[12pt]{article}
\usepackage{graphicx}
\title{An example document}
\author{Karl Broman}
\begin{document}
\maketitle
\thispagestyle{empty}
\section{A section}
This is a simple example of a \LaTeX\ document for an article. Here’s some in-line math: $y = \beta_0 + \beta_1 x + \epsilon$.
And here’s a display equation:
$$ \hat{\beta} = (X'X)^{-1} X'y $$
\end{document}

\LaTeX\ is like html or Markdown: plain text with special codes to indicate how things are to appear.

A \LaTeX\ document always starts with \documentclass, then a bunch of overall controlling information. The actual document is between \begin{document} and \end{document}.

\usepackage{} is like library() in R.

Ideally, you focus on semantics rather than style: define the \title{} and \author{} and use \maketitle to have them included in the document, and indicate sections and subsections with \section{} and \subsection{}.

For some reason, \thispagestyle{empty} ("don’t show page number on this page") needs to be placed after \maketitle.

A key feature of \LaTeX\ is the mathematics typesetting. There’s no better system. And your \LaTeX\ skills can be immediately transferred to your Markdown documents, with MathJax.
In reality, for a paper, I don’t use \maketitle or \section, but rather just muck about, hard-coding the placement of things.

But mine is not the recommended approach. If, for some reason, you need to change the style, it’s easier if your document is defined in terms of semantics.
Why \LaTeX?  

- Fine control of document appearance  
- Transparency of how that was achieved  
- Version control (diff/merge)  
- Typesetting equations  
- Markdown’s not quite ready, or sufficiently rich  
  (but see the R package rticles)

It’s a lot of work to learn \LaTeX, so we need to be clear about why we’d want to devote the effort to it.

For reproducible research, we need some sort of code-based document system (i.e., not Word!), and \LaTeX gives you the most fine-grained control, if you need it. Ultimately, I hope, Markdown will be sufficient, but for now, we often need \LaTeX.

The code-based control makes what you’re trying to do transparent. And you should treat \LaTeX like code: write clearly and simply, and comment the tricky bits.

This sort of document also has the advantage of easy treatment of `diff` and `merge` in a version control system like git.

The real power of \LaTeX is in the typesetting of mathematical equations. And what you learn on that aspect can be transferred to your Markdown documents, using MathJax. (But I already said that, didn’t I?)
\text{simple} \quad \longleftrightarrow \quad \text{flexible}

\centerline{\Large \text{simple} \quad \texttt{\$longleftarrow\$} \quad \texttt{\textbackslash quad flexible}}

\LaTeX\ sits at the right of the simple-to-flexible spectrum.
Modify your desires to match the defaults.

Focus your compulsive behavior on things that matter.

I’ve said this before, but I like to repeat it.

Focus on the text and the figures before worrying too much about fine details of how they appear on the page.

And consider which is more important: a manuscript, web page, blog, grant, course slides, course handout, report to collaborator, scientific poster.

You can spend a ton of time trying to get things to look just right. Ideally, you spend that time trying to construct a general solution. Or you can modify your desires to more closely match what you get without any effort.
These are bits of \LaTeX code that I use a lot.
KnitR works well with \LaTeX.

Most of what you learned about KnitR with R Markdown transfers directly to working with \LaTeX.

The main difference is the way in which code chunks are indicated. You use <<>>= and @ for chunks, and \texttt{\$Sexpr\$} for in-line code.

KnitR basically does a search-and-replace for code chunks. Different patterns will be easier, depending on the nature of the surrounding code.

The chunk options are the same. Here, I used \texttt{out.width="0.8\textwidth"} to make the figure appear as 80% of the width of the page.

\texttt{out.width} and \texttt{out.height} need units as in \LaTeX (built into \texttt{\textwidth}; otherwise "in" or "cm" or "pt" or whatever).

\texttt{fig.width} and \texttt{fig.height} are as in R, with implied units.
I create \LaTeX\ documents in emacs. If you want something WYSIWYG, consider LyX. KnitR is built in, and Yihui Xie strongly endorses it. (LyX is not really “WYSIWYG” but rather “WYSIWYM,” but that’s what you want most, anyway.)
There are a bunch of online tools for creating LaTeX documents, collaboratively.

I have no experience with these, but I’ve heard good things about Overleaf (formerly WriteLaTeX).
In addition to regular \LaTeX, there’s pdflatex (which I mostly use). It has the advantage of being able to include pdf, jpg, and png figures, and produces a PDF file directly.

XeLaTeX and LuaLaTeX are great for fonts and Unicode.

I’ve not mentioned that behind the scenes is \TeX, which is the source of all of this. Believe or not, \LaTeX exists because \TeX is even harder. PdfLaTeX, XeLaTeX, and LuoLaTeX, really derive from PdfTeX, XeTeX, and LuoTeX.
There is a ton of online information about LATEX. Start with google. It’s highly unlikely that you have a completely unique question or problem.

My last point here is basically that one way to help yourself is by learning to let things go.
Figure captions and floats

If you use the chunk option `fig.cap`, the figure will get a caption.

But it will also be embedded within a `figure` “environment.” (That is, between `\begin{figure}` and `\end{figure}.)

This makes it a “float.” LaTeX decides where it’s going to be placed. The placement of floats is the biggest pain in using LaTeX.

The figure also gets a label, from the chunk name. (The \label{} bit.) This allows you to cross-reference the figure, to have the figure number determined automatically.

The cross reference would be with \ref{fig:fig_with_caption}.

When you use cross references, you need to run LaTeX twice: once to establish where things will sit on the page and how they are numbered, and a second time to insert the cross references.
Tables in \LaTeX

\begin{tabular}{rrrrr} \hline
& Estimate & Std. Error & t value & Pr(>|t|) \\
(Intercept) & 0.04 & 0.11 & 0.4 & 0.69 \\
x & 0.98 & 0.10 & 10.0 & 0.00 \\
\hline
\end{tabular}

Tables in \LaTeX are a pain, but they offer extremely fine control.

But writing this sort of code (\& indicates breaks between columns, \\
indicates the end of a row) reproducibly is hard.
xtable

```r
<<generate_and_fit>>=
x <- rnorm(100)
y <- x + rnorm(100)
lm.out <- lm(y ~ x)
@

<<table, results="asis">>=
library(xtable)
xtable(lm.out, digits=c(0,2,2,1,2))
@

% a non-floating version
<<table, results="asis">>=
library(xtable)
xtab <- xtable(lm.out, digits=c(0,2,2,1,2))
print(xtab, floating=FALSE)
@
```

xtable is a superb R package for producing \LaTeX\ tables. You don't have complete control, but you do have a ton of control. The xtableGallery vignette shows you much of what can be done.

Note that a lot of the options are for `print.xtable`, so look at the help files for both `xtable` and `print.xtable`.

For example, if you don't want a table to be “floating,” (within a `table` environment, between \begin{table} and \end{table}), you need to use `print.table` with `floating=FALSE`.
Read proofs carefully

As submitted

\[ \Pr(g_1 = i, g_2 = j) = \begin{cases} \frac{1-r}{8(1+6r)} & \text{if } i = j \\ \frac{r}{8(1+6r)} & \text{if } i \neq j \end{cases} \]

As printed

\[ \Pr(g_1 = i, g_2 = j) = \begin{cases} \frac{1-r}{8(1+6r)} & \text{if } i = j \\ \frac{r}{2(1+6r)} & \text{if } i \neq j \end{cases} \]


Some journals re-type a bunch of your manuscript, sometimes introducing errors.

So read proofs carefully. (What pain!) And post a preprint, say to arXiv.org or bioRxiv.org.

The above is the most important equation in the paper, and I missed that they’d introduced a mistake.
I have a few papers with a lot of equations. I hope they’re not trying to re-type these. I generated them from code.
References with \LaTeX are via BibTeX, which is fabulous once you get used to it. Most software to track references will produce BibTeX files for you.

The formatting of citations and the reference listings, to match what the journal wants, can be painful. But I’ve figured out how to produce what Genetics wants, and I send all of my papers there.

The first box is the sort of code that would appear in your \LaTeX file: the bit at the top goes in the header (before $\texttt{\begin{document}}$). The bit in the middle shows how to cite papers: use $\texttt{\citep}$ to get the whole thing in parentheses, and use $\texttt{\citet}$ to get a reference like “...applied by Baggerly and Coombes (2008, 2009)...” The last bit in the first box produces the actual list of references.

The second box is the BibTeX format for a particular reference.

When you use BibTeX, you tend to run \texttt{pdflatex}, then \texttt{bibtex}, and then \texttt{pdflatex} a couple of more times.
Organizing analyses

- Directory for the main analysis project
  ~/Projects/Blah
- Directory for a paper
  ~/Docs/Papers/Blah
- Paper directory may have an analysis directory
  ~/Docs/Papers/Blah/Analysis
- Symbolic links to .RData files
  
  ln -s ~/Projects/Blah/DerivedData/blah.RData .

- Each part well organized and fully reproducible.
- R Markdown reports documenting different aspects.
- Analysis with the paper may be re-done "properly."

This is how I organize a paper related to a larger project.

Some of the work in the main project may be re-done a bit differently (or cleaner) in the analysis with the paper.

You don’t want to re-do all analyses for the paper, but it’d also be nice to have the data and code related to the paper be a bit more self-contained.

And usually when you’re sitting down to write the paper, you have better ideas about how to re-do things properly, and so it might be a good idea to go ahead and re-do things.

Ideally, you’d separate out each aspect of the analysis: data manipulation, data cleaning, and different parts of the analysis.

Have an R Markdown document describing each aspect, with the actual manuscript and its figures and tables drawing from the results of those R Markdown documents.
Every statistic, figure and table in your manuscript should be fully reproducible. So when you’re citing statistics, use \texttt{\textbackslash Sexpr{}} liberally.

This should inhibit you from writing numbers as words, though the \LaTeX{} code can get a bit ugly.

There’s a bit of fanciness here about capitalization and about ensuring that singular or plural nouns are correct. If \texttt{\textbackslash Sexpr{}} produces a character string, it ends up as plain text in your document.

I’ll use a lot of \texttt{myround()} from my R/broman package, too.

Long explanations or descriptions of figures can’t be fully reproducible, but the figures themselves and any statistics you mention should be.
While you could include all code in your .Rnw file, I prefer to pull out the code for my figures as separate files, and then write a Makefile for the manuscript construction and include them with \includegraphics.

The advantage of this is the ability to reuse the figures in talks or whatever. Also, journals will generally want the figures as separate files. Finally, the code for my figures is often incredibly long and ugly, so it’s best to separate it out.

Ideally, the code for a figure would be structured as a function and then a function call. Put a bit more effort into the code, so that you can reuse it later for a similar figure with different data. At the very least, you should write the repeated bits as functions.

If your function takes arguments that define the placement of things (padding for text and so forth), then the fine adjustments of the figure appearance would be easier.
Version Control

- Your manuscript is under version control, right?
- Local or private repository for the whole thing
  - including reviewers’ reports and my response
  - PDF of submitted and final manuscript
- Snapshot of the final version as a public repository
  - I don’t really want to show the whole history

Git is as good for tracking manuscripts and data analyses as it is for tracking code. Use it!

But I don’t want to make everything public, and I want to include private stuff in my repository.

I’ve been using just a local repository, but I’m moving towards having a private repository hosted on BitBucket.

I’ll put a snapshot of the final version, and maybe a few final changes, on GitHub.
With papers led by a collaborator, I’m usually stuck with Word.

But my analyses and figures are fully reproducible.

Create an R Markdown document with the detailed results.

Often, you’ll be stuck with Word. And you can’t reproducibly insert numbers into Word.

So have a separate R Markdown report with the detailed results, including every statistic that will get inserted into Word.

And take control of the figures and ensure that they are reproducible (and respectable).

Teach your collaborators to at least have their figures be reproducible?
Summary

- \LaTeX is brilliant for fine control and for equations
- Floating figures and tables can be a pain
- You use KnitR with \LaTeX much the same way as you’d used it with Markdown.
- Ensure that every statistic, figure, and table in your paper are fully reproducible.
- Use xtable to make tables.
- Separate out the code for the figures.
- Use version control!

Summaries are helpful.