Scripts define HOW

The report defines WHAT & WHY

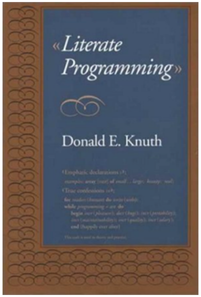
Mikhail Dozmorov

Fall 2016

Literate programming

Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, **let us concentrate rather on explaining to humans what we want the computer to do.**

–*Donald E. Knuth, Literate Programming, 1984*



knitR

Writing reports

- **HTML**: HyperText Markup Language, used to create web pages. Developed in 1993
- **LaTeX**: a typesetting system for production of technical/scientific documentation, PDF output. Developed in 1994
- **Sweave**: a tool that allows embedding of the R code in LaTeX documents, PDF output. Developed in 2002
- **Markdown**: a lightweight markup language for plain text formatting syntax. Easily converted to HTML

HTML example

- HTML files have .html extension
- Pairs of tags define content/formatting

```
<h1> Header level 1 </h1>
<a href="http://www.."> Link </a>
<p> Paragraph </p>
```

HTML example

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ut
</head>
<body>
<h1>Markdown example</h1>
<p>This is a simple example of a Markdown document.</p>
You can emphasize code with <strong>bold</strong> or <em>itali
</body>
</html>
```

LaTeX example

- LaTeX files usually have a .tex extension
- LaTeX commands define appearance of text, and other formatting structures

`http://www.electronics.oulu.fi/latex/examples/example_1`

LaTeX example

```
\documentclass{article}
\usepackage{graphicx}
\begin{document}
\title{Introduction to \LaTeX{}}
\author{Author's Name}
\maketitle
\begin{abstract}
This is abstract text: This simple document shows very basic f
\LaTeX{}```.
\end{abstract}
\section{Introduction}
```

Sweave example

- Sweave files typically have `.Rnw` extension
- LaTeX syntax for text, `<<chunk_name>>= <code> @` syntax outlines code blocks

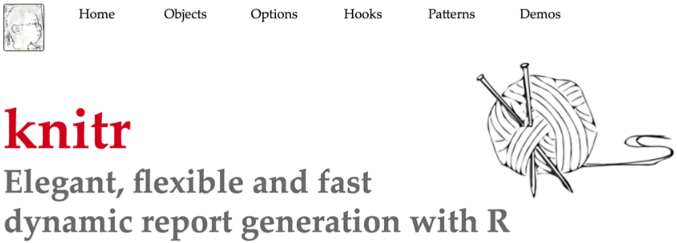
```
\documentclass{article}
\usepackage{amsmath}
\usepackage{natbib}
\usepackage{indentfirst}
\DeclareMathOperator{\logit}{logit}
% \VignetteIndexEntry{Logit-Normal GLMM Examples}
\begin{document}
First we attach the dat
<<booth>>=
library(bernor)
data(booth)
attach(booth)
@
```

KnitR

- KnitR: a package for dynamic report generation written in R Markdown. PDF, HTML, DOCX output. Developed in 2012

`https://github.com/yihui/knitr`

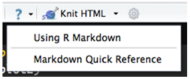
```
install.packages('knitr', dependencies = TRUE)
```



knitr

Elegant, flexible and fast dynamic report generation with R

Markdown syntax



`*italic* _italic_`	*italics*
`**bold** __bold__`	**bold**

Headers

```
# Header 1
## Header 2
### Header 3
```

Markdown syntax | Lists

Unordered List

```
* Item 1
* Item 2
  + Item 2a
  + Item 2b
```

Ordered List

```
1. Item 1
2. Item 2
3. Item 3
  + Item 3a
  + Item 3b
```

Markdown syntax

```
superscript^2^
~~strikethrough~~
```

Horizontal Rule / Page Break

```
******
------
```

Blockquotes

A friend once said:

```
> It's always better to give
> than to receive.
```

Markdown syntax

Links

```
http://example.com
[linked phrase](http://example.com)
```

Images

```
![](http://example.com/logo.png)
![optional caption text](figures/img.png)
```

Markdown syntax

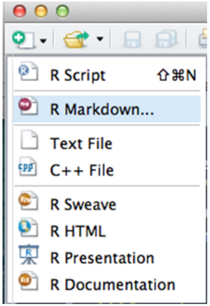
Tables

```
First Header  | Second Header
------------- | -------------
Content Cell  | Content Cell
Content Cell  | Content Cell
```

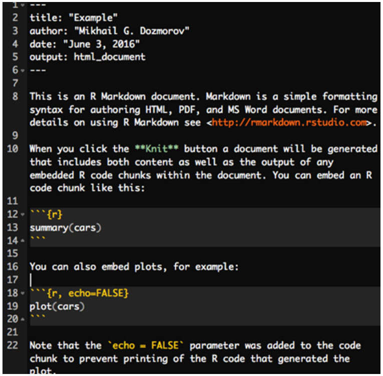
First Header	Second Header
Content Cell	Content Cell
Content Cell	Content Cell

Creating R markdown document

- Regular file with .Rmd extension
- Use RStudio



Creating R markdown document



~~~
---
title: "Example"
author: "Mikhail G. Dozmorov"
date: "June 3, 2016"
output: html_document
---

This is an R Markdown document. Markdown is a simple formatting
syntax for authoring HTML, PDF, and MS Word documents. For more
details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated
that includes both content as well as the output of any
embedded R code chunks within the document. You can embed an R
code chunk like this:

```{r}
summary(cars)
```

You can also embed plots, for example:

```{r, echo=FALSE}
plot(cars)
```

Note that the `echo = FALSE` parameter was added to the code
chunk to prevent printing of the R code that generated the
plot.
~~~

YAML header (think settings)

- YAML: YAML Ain't Markup Language
- YAML is a simple text-based format for specifying data, like JSON

```
---
title: "Untitled"
author: "Your Name"
date: "Current date"
output: html_document
---
```

`output` is the critical part - it defines the output format. Can be `pdf_document` or `word_document`

R Markdown | Code embedding

- Chunks of code are labeled

1. with single backticks, ‘<code>‘, rendered in a monospace font, non-executable. A simple code formatting option
2. with single backticks, ‘ r <code>‘, for inline code. **r** indicates executable R code. Instead of hard coding numbers, the inline code allows to evaluate variables in real time.

- There are ‘ r paste(nrow(my_data))‘ rows
- The estimated correlation is ‘ r cor(x, y)‘

`https://support.rstudio.com/hc/en-us/articles/205368677-R-Markdown-Dynamic-Documents-for-R`

Large code chunks

- Marked with triple backticks

~~~
```{r chunk_name, eval=FALSE}
x = Inf + .Machine$xmin
x
```
~~~

- The chunk name is optional
- By default, the code AND its output are displayed in the final report

Chunk options, comma-separated

- `echo=FALSE` (Default: TRUE): hides the code, but not the results/output.
- `results='hide'` (Default: 'asis') hides the results/output. 'hold' - hold all the output until the end of a chunk.
- `eval=FALSE` (Default: TRUE): disables code execution.
- `cache=TRUE` (Default: FALSE): turn on caching of calculation-intensive chunk.
- `fig.width=##`, `fig.height=##`: customize the size of a figure generated by the code chunk

Global chunk options

- Some options you would like to set globally, instead of typing them for each chunk

~~~
```{r global_options, eval=FALSE}
knitr::opts_chunk$set(fig.width=12, fig.height=8, fig.path='Fi
                      echo=FALSE, warning=FALSE, message=FALSE
```
~~~

- `warning=FALSE` and `message=FALSE` suppress any R warnings or messages from being included in the final document
- `fig.path='Figs/'` the figure files get placed in the Figs subdirectory. (Default: not saved at all)

`https://github.com/mdozmorov/MDmisc`

An example of R Markdown document

~~~
```{r libraries, echo=TRUE}
library(ggplot2)
```
~~~

There are ‘ r paste(length(LETTERS))‘ letters in English alphabet.

~~~
```{r count_combinations, echo=TRUE}
max_number_of_combinations <- 5
count_combinations <- list()
for (i in 1:max_number_of_combinations) {
  count_combinations <- c(count_combinations, ncol(combn(lengt
}
```
~~~

A total of ‘ r paste(count_combinations[[2]])‘ pairwise combinations of them can be selected. Or, ‘ r paste(count_combinations[[3]]) ‘ combinations of three letters can be selected.

Displaying data as tables

- knitR has built-in function to display a table

```
data(mtcars)
knitr::kable(head(mtcars))
```

- pander package allows more customization

```
pander::pander(head(mtcars))
```

- xtable package has even more options

```
xtable::xtable(head(mtcars))
```

- DT package, an R interface to the DataTables library

```
DT::datatable(mtcars)
```

Creating the final report

- Markdown documents `*.md` can be converted to HTML using

```
markdown::markdownToHTML('markdown_example.md',
'markdown_example.html')
```

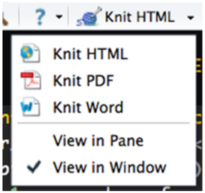
- Another option is to use:

```
rmarkdown::render('markdown_example.md')
```

At the backend it uses pandoc command line tool, installed with Rstudio
`http://pandoc.org/`

Creating the final report

- Rstudio: one button
- `knit2html()`, `knit2pdf`



- **Note**: KnitR compiles the document in an R environment separate from yours (think Makefile). Do not use **./Rprofile** file.

Things to include in your final report

~~~
```{r session_info, results='hide', message=FALSE}
library("dplyr")
library("pander")
diagnostics <- devtools::session_info()
platform <- data.frame(diagnostics$platform %>% unlist, strin
colnames(platform) <- c("description")
pander(platform)
packages <- as.data.frame(diagnostics$packages)
pander(packages[ packages$`*` == "*", ])
```
~~~

- Include `session_info()` at the end: outputs all packages/versions used
- `set.seed(12345)`: initialize random number generator