RAdelaide 2015:
Writing Reports Using R Markdown

Steve Pederson
Bioinformatics Hub
University of Adelaide

Wednesday August 19th, 2015

Welcome To Day 2

Today's Topics

1. **Writing Reports Using `rmarkdown`**
2. Data Types in *R*
3. Writing Functions In *R*
4. Parallel Processing in *R*

Writing Reports Using `rmarkdown`

Now that we know how to load & tidy data:

- `rmarkdown` is a cohesive way to
  - Load & tidy data
  - Analyse data, including figures & tables
  - Publish everything in a complete report/analysis

We can do this all in one document, with our analysis code embedded!

Writing Reports Using `rmarkdown`

Using `rmarkdown` we can output our analysis directly as

- HTML
- MS Word Documents
- PDF Documents (like this one)
- Slidy or `ioslides` presentations

We never need to use MS Word or Excel ever again!!!

Writing Reports Using `rmarkdown`

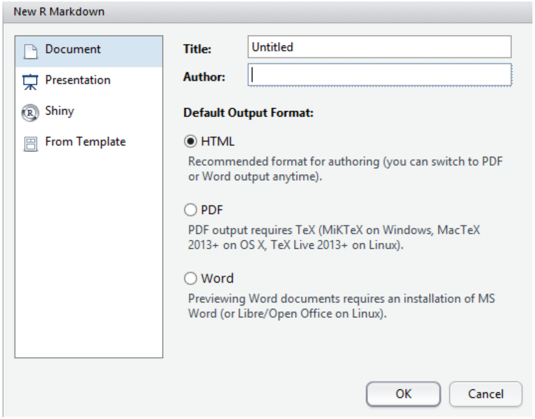
- The file format ends with `.Rmd` and allows us to include normal text alongside embedded *R* code.
- We can create all of our figures & tables directly from the raw data, as well as
  - Data or experimental descriptions
  - Descriptions of every stage of the analysis
  - Analytic conclusions
  - Any other information
- We can easily format text and include complex equations along with our code.

Creating an *R* Markdown document

Let's create our first `rmarkdown` document

1. Ensure you have `rmarkdown` installed
2. Go to the `File` drop-down menu in RStudio
3. New File -> R Markdown...

The New R Markdown Form



Title: Untitled

Author:

Default Output Format:

[✓ HTML]
Recommended format for authoring (you can switch to PDF or Word output anytime).

PDF
PDF output requires TeX (MiKTeX on Windows, MacTeX 2013+ on OS X, TeX Live 2013+ on Linux).

Word
Previewing Word documents requires an installation of MS Word (or Libre/Open Office on Linux).

The New R Markdown Form

1. Change the Title to: My First Report
2. Change the Author to your own name
3. Leave everything else as it is & hit OK
4. Save the file as `RMarkdownTutorial.Rmd`

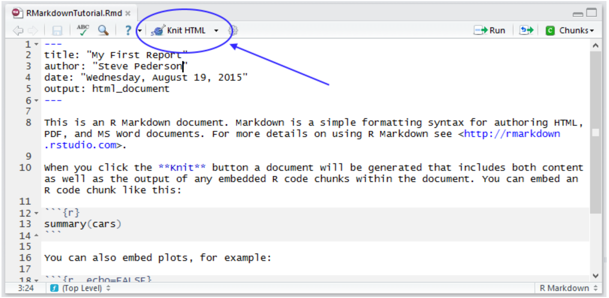
The R Markdown Format

Looking at the file:

- The header section is contained within the two --- delineation lines
- Lines 8 – 10 are plain text
    - Bold is indicated by **Knit**
    - Italics can be indicated using a single *
- Lines 12 – 14 are an `R` code `chunk`
    - Chunks always begin with ```{r}
    - Chunks always end with ```
    - The code goes between these two delineators

The R Markdown Format

- The default format is an `html_document` & we can change this later.
- Generate the document by clicking `Knit HTML`



~~~
---
title: "My First Report"
author: "Steve Pederson"
date: "Wednesday, August 19, 2015"
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}
```

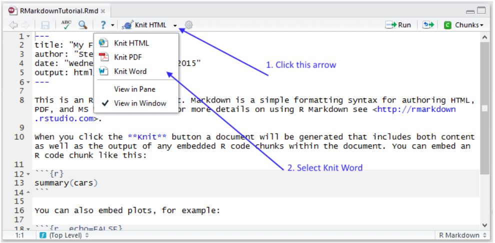
The R Markdown Format

A preview window will appear with the compiled report

- Note the hyperlink to the RMarkdown website & the bold typeface for the word **Knit**
- The *R* code and the results are printed for the line `summary(cars)`
- The plot of `speed` Vs. ‘distance has been embedded
- The code generating the plot was hidden using `echo = FALSE`

The R Markdown Format

We could also save this as an MS Word document (i.e. `.docx` format)



~~~
---
title: "My F
author: "Ste
date: "Wedne                    2015"
output: html
---

This is an R                t. Markdown is a simple formatting syntax for authoring HTML,
PDF, and MS                 or 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}
```

By default, this will be Read-Only

Saving as a `.PDF` may require an installation of $\LaTeX$.

Making our own report

Now we can modify the code to create our own analysis.

- Delete everything in your R Markdown file EXCEPT the header
- We'll analyse a dataset called `ToothGrowth` contained in the *R* package `datasets`
- First we'll need to describe the data

```
?ToothGrowth
```

Describing the data

Rename the report

First we should change the title of the report to something suitable, e.g. *The Effects of Vitamin C Delivery Methods on Tooth Growth*

Create a “Data Description" Section

Now let’s add a section header for our analysis to start the report

1. Type `# Data Description` after the header and leaving a line
2. Use your own words to describe the data

Describing the data

My text was:

We are interested in the effects of Vitamin C on tooth growth, and for this experiment we used Guinea Pigs as the model organism. We gave guinea pigs Vitamin C supplements at three doses (0.5, 1 and 2mg) using two delivery methods. Delivery method 1 was orange juice, whilst delivery method 2 was ascorbic acid. Each treatment combination was given to 10 guinea pigs, giving a total sample size of 60.

The measured response was the length of odontoblasts.

Describing the data

Hopefully you mentioned that there were 10 guinea pigs in each group, with a total of 60.

Question

Can we get that information from the data itself?

Describing the data

Hopefully you mentioned that there were 10 guinea pigs in each group, with a total of 60.

Question

Can we get that information from the data itself? **Yes**

Solution

`nrow(ToothGrowth)` would give the total number of samples

Describing the data

We can actually embed this in our data description!

1. Instead of the number 60 in your description, enter:

   ```
   ` r nrow(ToothGrowth) `
   ```

2. Recompile the HTML document.

Adding some *R* code

After our description, we could have a look at the data in a summary

```
ToothGrowth %>%
  group_by(supp, dose) %>%
  summarise(Count = n())
```

However, we'll need to add a code chunk before this to load the package `dplyr`

Loading *R* packages

1. Before the Data Description header, add a new header called `Required Packages`
2. Create a code chunk with the contents `library(dplyr)`
3. Recompile the HTML

Adding this has loaded the package `dplyr` so that any code chunks following will be able to use the functions in the package

Tidying our output

Notice that loading `dplyr` gave us an overly informative message.

We can turn this off to make the report look nicer

1. After the `r` inside the brackets at the start of the code chunk, add a comma
2. Start typing the word `message` and use the auto-complete feature to set `message = FALSE`
3. Recompile

Add the data summary

```r
ToothGrowth %>%
  group_by(supp, dose) %>%
  summarise(Count = n())
```

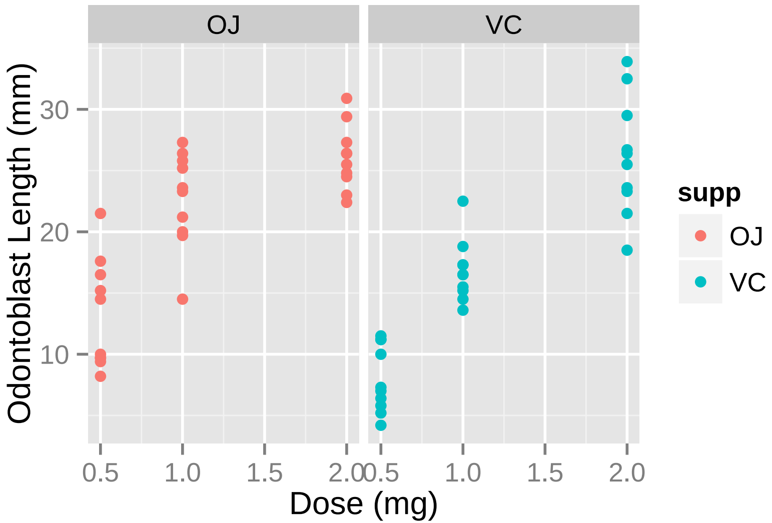
```
## Source: local data frame [6 x 3]
## Groups: supp
##
##   supp dose Count
## 1   OJ  0.5    10
## 2   OJ  1.0    10
## 3   OJ  2.0    10
## 4   VC  0.5    10
## 5   VC  1.0    10
## 6   VC  2.0    10
```

Add a plot of the data

We can use `ggplot2` for this

1. Load this package back in the **Required Packages** section
2. Create a plot using `geom_point()`
3. Colour the points based on the `supp` variable
4. Facet the plot based on the `supp` variable

Add a plot of the data



Analyse the data

One way to analyse this data would be to conduct a *t*-test at each dosage level

1. Create a new section called `Analysis`
2. Create a lower level heading called `Dosage == 0.5`.
   (This can be done using 2 or 3 hashes)
3. Create a code chunk with the following code

```
(lowDosage <- t.test(len~supp, subset(ToothGrowth, dose==0.5)))
```

Analyse the data

Questions

1. Why would we store the results as an *R* object?
2. What do the round brackets at the start & end of the code do?

Analyse the data

1. Repeat the above for the medium and high dosage levels
2. Write some kind of conclusion based on the above results
3. To embed the *p*-values directly in our text, we can call the *R* objects we've created via the embedded code procedure `` `r embedded code` ``

Hints

1. The *p*-values can be accessed via `lowDosage$p.value`
2. We may also like to restrict the number of decimal places using the command `round()`

If we have time

We could provide an even more informative output by plotting confidence intervals

1. Create a new subsection called `Confidence Intervals`
2. Create a `data.frame` with variables `dose`, `diff`, `lower` & `upper` filling with values from the *R* objects we've already created
3. Plot these intervals using`geom_point` and `geom_errorbar`

Hints

1. The estimates of the mean tooth growth will be in the *R* objects in a component called `$estimate`
2. The lower and upper bounds for the 95% confidence interval will be in the component called `$conf.int`

Finishing the analysis

After you're happy with the way your analysis looks

- A good habit is to finish with a section called `Session Info`
- Add a code chunk which calls the *R* command `sessionInfo()`

Finishing the analysis

So far we've been compiling everything as HTML, but let's switch to an MS Word document

We could email this to our supervisors, or upload to Google docs for collaborators. . .

Summary

This basic process is incredibly useful

- We never need to cut & paste anything between document formats
- Every piece of information comes directly from our *R* analysis
- We can very easily incorporate new data
- Analyses can be simply changed as new information comes to hand
- Creates *reproducible research*