Reproducible research 1

More with knitr

Equations in `knitr`

You can write equations in RMarkdown documents by setting them apart with dollar signs ($). For an equation on a line by itself (**display equation**), you two $s before and after the equation, on separate lines, then use LaTex syntax for writing the equations.

To help with this, you may want to use this LaTex math cheat sheet.. You may also find an online LaTex equation editor like Codecogs.com helpful.

Note: Equations denoted this way will always compile for pdf documents, but won't always come through on Markdown files (for example, GitHub won't compile these math).

Equations in `knitr`

For example, writing this in your R Markdown file:

```
$$
E(Y_{t}) \sim \beta_{0} + \beta_{1}X_{1}
$$
```

will result in this rendered equation:

$$E(Y_t) \sim \beta_0 + \beta_1 X_1$$

Equations in `knitr`

To put math within a sentence (**inline equation**), just use one $ on either side of the math. For example, writing this in a R Markdown file:

```
"We are trying to model $E(Y_{t})$."
```

The rendered document will show up as:

"We are trying to model $E(Y_t)$."

Figures

You can include not only figures that you create with R, but also figures that you have saved on your computer.

The best way to do that is with the `include_graphics` function in `knitr`:

```
library(knitr)
include_graphics("../figures/CSU_ram.png")
```

Figures

```
library(knitr)
include_graphics("../figures/CSU_ram.png")
```

This example would include a figure with the filename "MyFigure.png" that is saved in the "figures" sub-directory of the parent directory of the directory where your .Rmd is saved.

Don't forget that you will need to give an absolute pathway or the relative pathway **from the directory where the .Rmd file is saved**.

Saving graphics files

You can save figures that you create in R. Typically, you won't need to save figures for an R Markdown file, since you can include figure code directly.

However, you will sometimes want to save a figure from a script. You have two options:

- Use the "Export" choice in RStudio
- Write code to export the figure in your R script

To make your research more reproducible, use the second choice.

Saving graphics files

To use code to export a figure you created in R, take three steps:

1. Open a graphics device (e.g., `pdf("MyFile.pdf")`).
2. Write the code to print your plot.
3. Close the graphics device using `dev.off()`.

Saving graphics files

For example, the following code would save a scatterplot of time versus passes as a pdf named "MyFigure" in the "figures" subdirectory of the current working directory:

```
pdf("figures/MyFigure.pdf", width = 8, height = 6)
ggplot(worldcup, aes(x = Time, y = Passes)) +
        geom_point(aes(color = Position)) +
        theme_bw()
dev.off()
```

If you create multiple plots before you close the device, they'll all save to different pages of the same pdf file.

Saving graphics files

You can open a number of different graphics devices. Here are some of the functions you can use to open graphics devices:

- `pdf`
- `png`
- `bmp`
- `jpeg`
- `tiff`
- `svg`

Saving graphics files

You will use a device-specific function to open a graphics device (e.g., `pdf`). However, you will always close these devices with `dev.off`.

Most of the functions to open graphics devices include parameters like `height` and `width`. These can be used to specify the size of the output figure. The units for these depend on the device (e.g., inches for `pdf`, pixels by default for `png`). Use the helpfile for the function to determine these details.

Saving graphics files

The `ggsave` function from the `ggplot2` package can also be used to save plots to files. It allows you to specify the graphics device within the `ggsave` function call.

kable

If you want to create a nice, formatted table from an R dataframe, you can do that using `kable` from the `knitr` package.

```
my.df <- data.frame(letters = c("a", "b", "c"),
                    numbers = 1:3)
kable(my.df)
```

letters	numbers
a	1
b	2
c	3

kable

There are a few options for the `kable` function:

arg	expl
`colnames`	Column names (default: column name in the dataframe)
`align`	A vector giving the alignment for each column ('l', 'c', 'r')
`caption`	Table caption
`digits`	Number of digits to round to. If you want to round columns different amounts, use a vector with one element for each column.

kable

```
my.df <- data.frame(letters = c("a", "b", "c"),
                    numbers = rnorm(3))
kable(my.df, digits = 2, align = c("r", "c"),
      caption = "My new table",
      col.names = c("First 3 letters",
                    "First 3 numbers"))
```

Table 3: My new table

First 3 letters	First 3 numbers
a	1.05
b	0.84
c	1.26

`xtable`, `pander`, `kableExtra`

From Yihui:

> "***Want more features?*** *No, that is all I have. You should turn to other packages for help. I'm not going to reinvent their wheels.*"

If you want to do fancier tables, you may want to explore the `xtable`, `pander`, and `kableExtra` packages.

Sometimes these will operate differently when compiling to pdf versus HTML.