Open, transparent, and reproducible research with RMarkdown, Jupyter notebooks, and Git

Mark Andrews
Psychology Department, Nottingham Trent University

mark.andrews@ntu.ac.uk
@xmjandrews
https://github.com/lawsofthought/research-integrity-2017

October 23, 2017

How we usually work

When carrying out computing/data-analysis intensive scientific research, the following is the common *modus operandi*:

- Interactive processing, analysis, and visualization of data.
- Copying-and-pasting values and figures into reports.
- The report, and not the data and code, is then made public.

Problems with the traditional approach

- Interactive work, followed by copying-and-pasting results is inherently error prone.
- It is also highly inefficient; even small changes become prohibitively expensive.
- The workflow is not reproducible; the details of the pipeline from raw data to reported results are not recorded.
- The reported results are not transparent; the public views only a carefully selected facade.
- Data and code are separated from the report and remain hidden. Data and code are the second class citizens of scientific communication.

Doing open, reproducible, and transparent analysis

The following are some of the tools that can greatly facilitate open, reproducible, and transparent analysis:

- RMarkdown
- Knitr
- pandoc
- packrat
- Jupyter notebooks
- pip & virtual environments
- Git
- GitHub
- Git fat, Git annex, Git LFS
- VirtualBox, Docker, Vagrant, etc.

RMarkdown: Example 1

We write source code that is mixture of R code and explanatory text that optionally references the R variables.

````
```{r}
set.seed(101)
N <- 50
mu <- 100
sigma <- 15
x <- rnorm(N, mean=mu, sd=sigma)
```

The mean of a random sample of `r N` numbers,
drawn independently from a normal distribution
with mean `r mu` and standard deviation `r sigma`,
is `r round(mean(x), 2)`.
````

RMarkdown: Example 1 (rendered)

When we render this, we'll produce a document (in this case, LaTeX) with both the code and any output and any evaluated variables in the text.

```
set.seed(101)
N <- 50
mu <- 100
sigma <- 15
x <- rnorm(N, mean=mu, sd=sigma)
```

The mean of a random sample of 50 numbers, drawn independently from a normal distribution with mean 100 and standard deviation 15, is 98.14.

RMarkdown: Example 2

We may turn off the rendering of the R source code with `echo = FALSE`.

````
```{r, echo=FALSE}
set.seed(101)
N <- 50
mu <- 100
sigma <- 15
x <- rnorm(N, mean=mu, sd=sigma)
```

The mean of a random sample of `r N` numbers,
drawn independently from a normal distribution
with mean `r mu` and standard deviation `r sigma`,
is `r round(mean(x), 2)`.
````

RMarkdown: Example 2 (rendered)

Then we get e.g. just the rendered text, but not the R *chunk*.

The mean of a random sample of 50 numbers, drawn independently from a normal distribution with mean 100 and standard deviation 15, is 98.14.

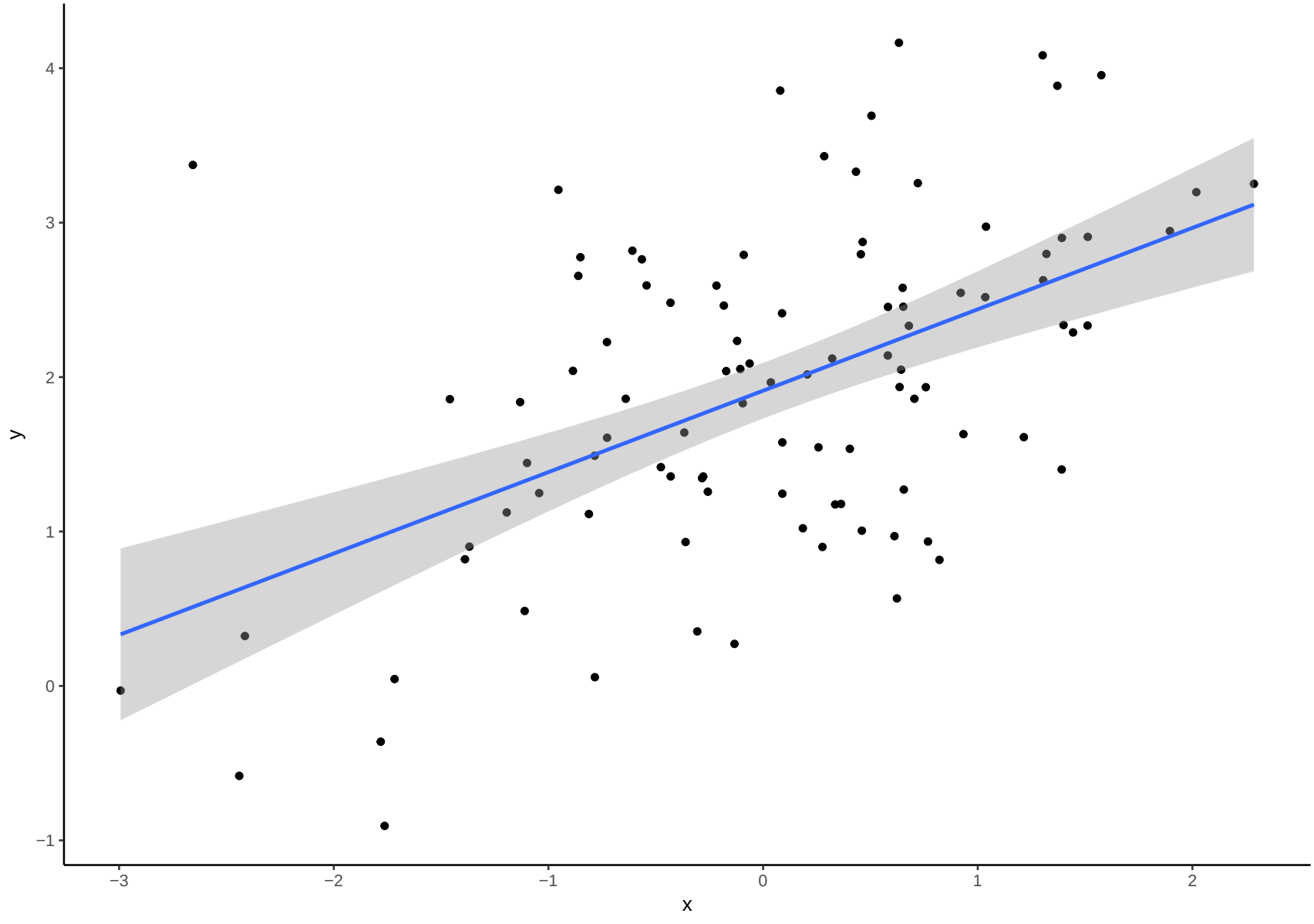
RMarkdown: Example 3

Figures will be rendered and inserted into the document in an identical manner.

````
```{r, echo=FALSE}
set.seed(42)
N <- 100
x <- rnorm(N)
Df <- data.frame(x = x,
                 y = 2 + 0.5*x + rnorm(N))
ggplot(Df,
       mapping = aes(x=x, y=y)) +
  geom_point() +
  stat_smooth(method='lm') +
  theme_classic()

```
````

RMarkdown: Example 3 (rendered)



RMarkdown: Example 4

Likewise, tables from statistical models can be rendered and inserted into the document.

````
```{r, echo=FALSE}
set.seed(42)
N <- 100
x <- rnorm(N)
Df <- data.frame(x = x,
                 y = 0.0 + 0.25*x + rnorm(N))

M <- lm(y ~ x, data=Df)
pander(summary(M))
```
````

RMarkdown: Example 4 (rendered)

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.088	0.091	-0.972	0.333
x	0.277	0.088	3.162	0.002

Table 2: Fitting linear model: y ~ x

Observations	Residual Std. Error	R^2	Adjusted R^2
100	0.908	0.093	0.083

RMarkdown: Example 5

RMarkdown allows us to typeset mathematical equations, symbols, etc., just as we would do with LaTeX.

````
```{r, echo=FALSE}
set.seed(42)
N <- 100
x <- rnorm(N)
Df <- data.frame(x = x,
                 y = 0.0 + 0.25*x + rnorm(N))
M <- lm(y ~ x, data=Df)
```


The linear model is
$$
y_i = \alpha + \beta x_i + \epsilon_i,
\quad \text{for $i \in 1 \ldots N$}.
$$

The $R^2$ value is `r round(mean(summary(M)$r.sq),2)`.
````

RMarkdown: Example 4 (rendered)

The linear model is

$$y_i = \alpha + \beta x_i + \epsilon_i, \quad \text{for } i \in 1 \dots N.$$

The R^2 value is 0.09.

RMarkdown, knitr, and pandoc

- RMarkdown is basically a combination of a *Markdown* document with embedded R code that is evaluated by `knitr`.
- The resulting file is processed by pandoc to produce the desired output file type.
- The three main output options are
  - pdf from LaTeX
  - MS Word
  - HTML
- knitr can also be combined directly with LaTeX, and this allows even more control of the final document.

Jupyter notebooks

- Jupyter notebook are browser-based dynamic documents.
- They are ideal for interactive work, prototype code development, visualization.
- They generate rendered documents, e.g. pdf via LaTeX, html, etc, like RMarkdown.
- Jupyter notebooks support multiple languages, but began as an extension of the IPython project, and Python is probably the most widely used language.

See demo available at `https://try.jupyter.org`

Git & GitHub

- Git is version control software, initially developed for version control of the Linux operating system kernel.
- It is now extremely widely used for almost all kinds of software development projects.
- Git works on a decentralized system whereby a code-base can be *cloned*, developed independently, and possible re-merged.
- For collaborating on one project, two developers use a *remote* host, clone it, develop locally, *commit* and then *push* back to and *pull* from the remote host.
- GitHub is one of the most widely used hosting sites (but there are others, e.g. BitBucket; and running your own git hosting server is simple and inexpensive).
- Git is designed for source code (i.e. text files) management. Data and other "assets" can be attached to (rather than kept within) the repository using `git fat`, `git annex`, `git lfs`, etc.

Git: Tiny tutorial

- Start by cloning a remote repository:

```
git clone https://github.com/yihui/knitr.git
cd knitr
git log # Read all the commit logs
```

- Work as normal, i.e. edit files, create new files, delete files.
- You now *stage* your changes, e.g.

```
git add foo.file.1 foo.file.2 # for edits or new files
git rm foo.file.3 # for removed files
```

- You then *commit* these:

```
git commit # Editor opens for your log msg
```

Git: Tiny tutorial (2)

- Pull down any recent changes by others from the remote:

```
git pull
git log # If new changes, read their logs
```

- Now, push your own changes to the remote

```
git push # requires permissions
```

- Undo changes:

```
git reset a381f2f # move back "head"
git revert a381f2f # applies new change to revert
```

Conclusions

- For open, transparent, and reproducible analysis, the code, data, and explanatory texts must be remain coupled.
- RMarkdown (and knitr) and Jupyter notebooks are two complementary approaches to coupling data, code, and explanatory text.
- Git & GitHub allows us to manage, develop, and share these interdependent files.