Dynamic Documents For Your Research Workflow

Fernando Hoces de la Guardia
BITSS

-

Slides at `https://goo.gl/ZFQvba`

BITSS Annual Meeting, December 2017

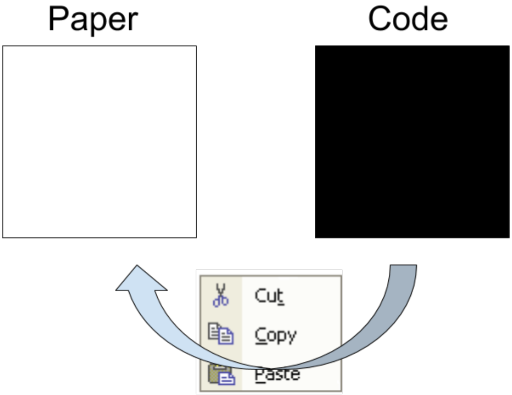
Dynamic Documents For Computational Reproducibility

First: can you take this quick survey please?

Dynamic Documents For Computational Reproducibility

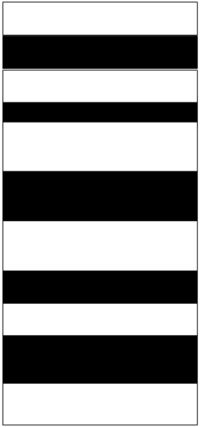
- Based on principles of *literate programming* aims at combining code and paper in one single document
- Best framework to achieve the holy grail of **one-click reproducible workflow**
- Best two current implementations: `RMarkdown (R)` & `Jupyter (Python)`. `Stata` is catching up (more at the end)

Currently code and narrative components live in separate universes



Dynamic Documents: integrate the two universes!

Paper + Code



Dynamic Documents: A Reciepe

- 1 simple language that can combine text and code: `Markdown`
- 1 statistical package to do the analysis (`R`, `Python`, `3S's?`)
- 1 machinery to combine analysis and text to create a single output: `Pandoc`
- [Optional-but-not-really] 1 program to bring all the elements together: RStudio/RMarkdown, Jupyter

One Type of Dynamic Document: R Markdown

For our excercise: R Markdown

- R: **open source** programming language design for statistical analysis.
- RStudio: free software that provides and Integrated Development Environment (IDE)
- RStudio combines all together: R + Markdown + Pandoc to produce multiple outputs

R Markdown


Bookdown
PDF
LaTeX
TUFTE
HTML
Slide
Headline One
Headline Two

Basic Structure

- A header
- Text
- Code: inline and chunks

Basic Structure: Header

```
---
title: "Sample Paper"
author: "Fernando Hoces de la Guardia"
output: html_document
---
```

Basic Structure: Body of Text

```
---
header
---
```

This is where you write your paper. Nothing much to add. You can check Markdown syntax here. And it can use can type equations using LaTex syntax!

Basic Structure: Code Chunks and Inline

```
---
header
---
```

Body of text.

To begin a piece of code ("code chunk"). Enclose them in the following expression (Ctrl/Cmd + shift/optn + i)

```
```{r, eval=TRUE}
here goes the code
```
```

To write inline use only one Backtick to open followed by an "r"" and one to close ‘r 1+1‘ in the output.

Practical Excercise

Hands-on excercise: the birthday problem!

As an illustration lets write a report using the participants in this workshop to illustrate the famous birthday problem.

> *What is the probability that at least two people this room share the same birthday?*
>
> *Is it something like* $\frac{1}{365} \times N =$ *0.11?*

Create a new RMarkdown File

1 - In RStudio: `File-> New File -> RMarkdown...`
2 - Name it, and save it.
3 - Review/edit the header, and delete all the default body of text except for one code chunk.
4 - Define a seed (`set.seed = 1234` and number of people in the room (`n.pers = ?`)

The birthday problem: the math

Actually the math says otherwise:

$$
\begin{aligned}
1-\bar{p}(n) &= 1 \times \left(1-\frac{1}{365}\right) \times \left(1-\frac{2}{365}\right) \times \cdots \times \left(1-\frac{n-1}{365}\right) \\
&= \frac{365 \times 364 \times \cdots \times (365-n+1)}{365^n} \\
&= \frac{365!}{365^n(365-n)!} = \frac{n! \cdot \binom{365}{n}}{365^n} \qquad (1) \\
p(n=40) &= 0.891
\end{aligned}
$$

Code for the math (https://goo.gl/ZFQvba)

Don't look at this: just copy and paste into your report

```
\begin{align}
 1 - \bar p(n) &= 1 \times \left(1-\frac{1}{365}\right)
 \times \left(1-\frac{2}{365}\right) \times \cdots \times
 \left(1-\frac{n-1}{365}\right) \nonumber  \\
 &= \frac{ 365 \times 364 \times \cdots \times
   (365-n+1) }{ 365^n } \nonumber \\
 &= \frac{ 365! }{ 365^n (365-n)!} =
   \frac{n!\cdot\binom{365}{n}}{365^n}\\
p(n= `r n.pers`) &= `r
 round(1 - factorial(n.pers) *
         choose(365,n.pers)/ 365^n.pers, 3)`\nonumber
\end{align}
```

Don't like math? Let's run a simple simulation!

1 - Simulate 10,000 rooms with $n = 40$ random birthdays, and store the results in matrix where each row represents a room.
2 - For each room (row) compute the number of unique birthdays.
3 - Compute the average number of times a room has 40 unique birthdays, across 10,000 simulations, and report the complement.

Code for the simulation (https://goo.gl/ZFQvba)

```
birthday.prob = function(n.pers, n.sims) {
  # simulate birthdays
  birthdays = matrix(round(runif(n.pers * n.sims, 1, 365))
                     nrow = n.sims, ncol = n.pers)
  # for each room (row) get unique birthdays
  unique.birthdays = apply(birthdays, 1, unique)
  # Indicator with 1 if all are unique birthdays
  all.different = (lapply(unique.birthdays, length) == n.pe
  # Compute average time all have different birthdays
  result = 1 - mean(all.different)
return(result)
}
n.pers.param = 43
n.sims.param = 1e4
birthday.prob(n.pers.param,n.sims.param)
```

```
## [1] 0.9216
```

Results

- Many people originally think of a prob $\sim \frac{1}{365} \times N = 0.118$
- However the true probability is of $p(n = 43) = 0.924$
- And the simulated probability is of 0.926

Final Remarks & More Resources

Final Remarks & More Resources

- With DD with can achieve a one-click reproducible workflow.
- This is particularly helpful to understand/present results that are hard to digest.
- Stata just develop an internal version of DD for v15. Review Here
- More great examples here
- Want to learn more: great free books (can you guess how they were written?)