Reproducible Workflow: Software

Quick Overviews of Version Control (Github) and Dynamic Documents (RMarkdown)

Fernando Hoces de la Guardia
BITSS

-

Slides at https://goo.gl/aBQ3LR

Inter-American Development Bank Workshop, March 2018

Version Control

Dynamic Documents

Version Control

Version Control Problem to avoid

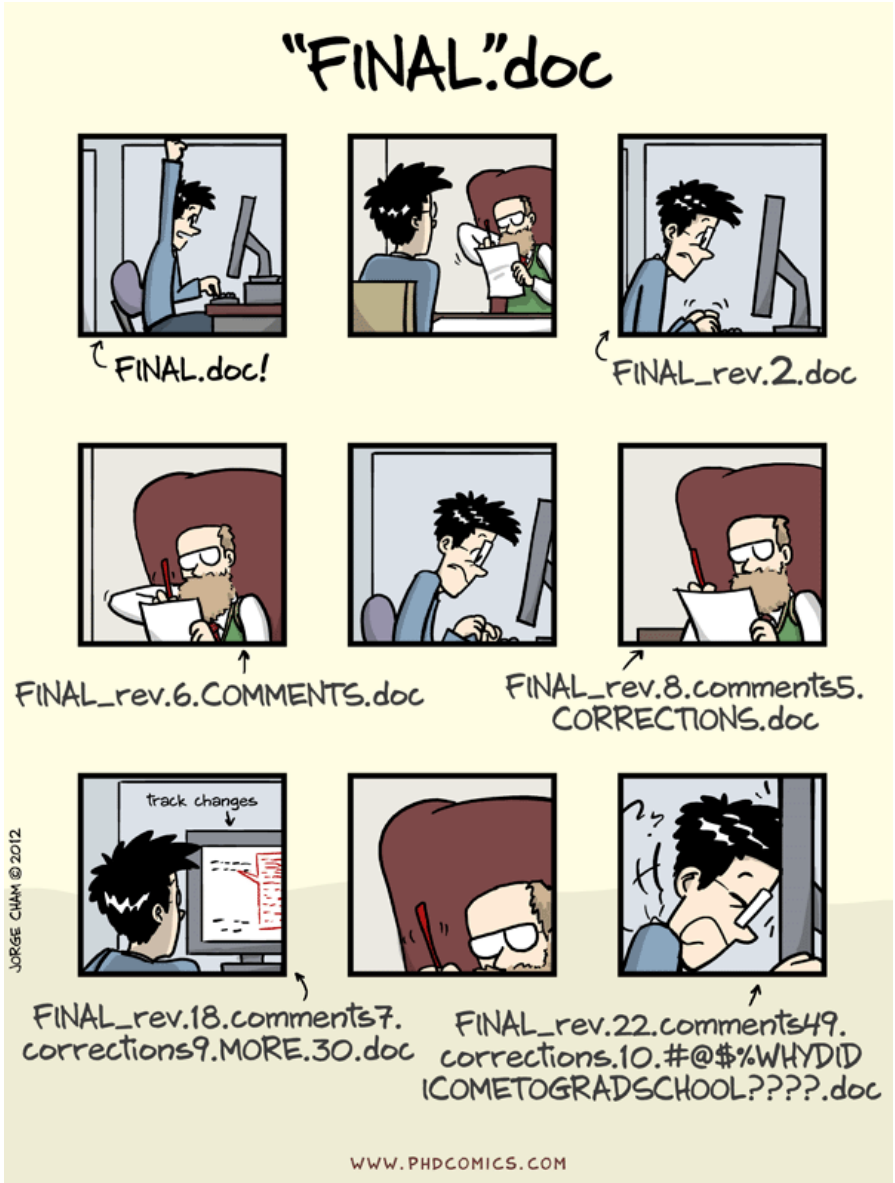


Figure 1: http://www.phdcomics.com/comics/archive/phd101212s.gif

Managing expectations

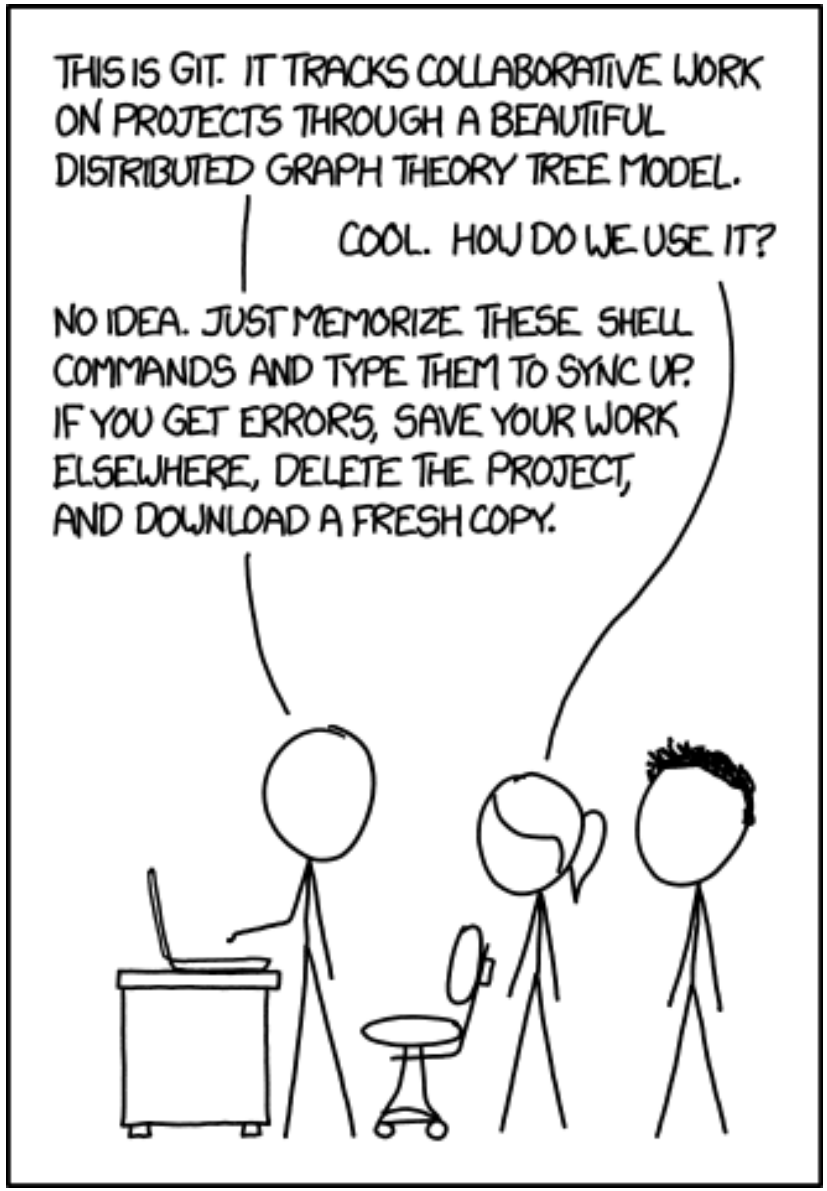


Figure 2: Git xkcd comic

The Primary Goal of Version Control (for us)

The Goal: keep track of any potentially meaningfull modification to your code (Git is also great for collaboration and exploring the work of others).

Strategy 1:

1 - Agree on a naming convention with you co-authors (eg: YYYYMMDDfilename_INITALS).
2 - Begin working from the last saved version (eg: `20180325demo_FH.do`).
3 - At the end of the day, save on a new version (eg: `20180327demo_FH.do`).

Pros: Easy adoption.

Cons: Error prone, hard to document, lots of files for each document.

Strategy 2:

1 - Name your file `filename` (ideally `01_filename`)
2 - Take a snapshot of your work every time you complete relevant change (day, hour or minutes).
3 - Update your entire working folder to the cloud.

Pros: Error proof, seemless documentation, one file per document, track differences across all versions, meant to work with the cloud.

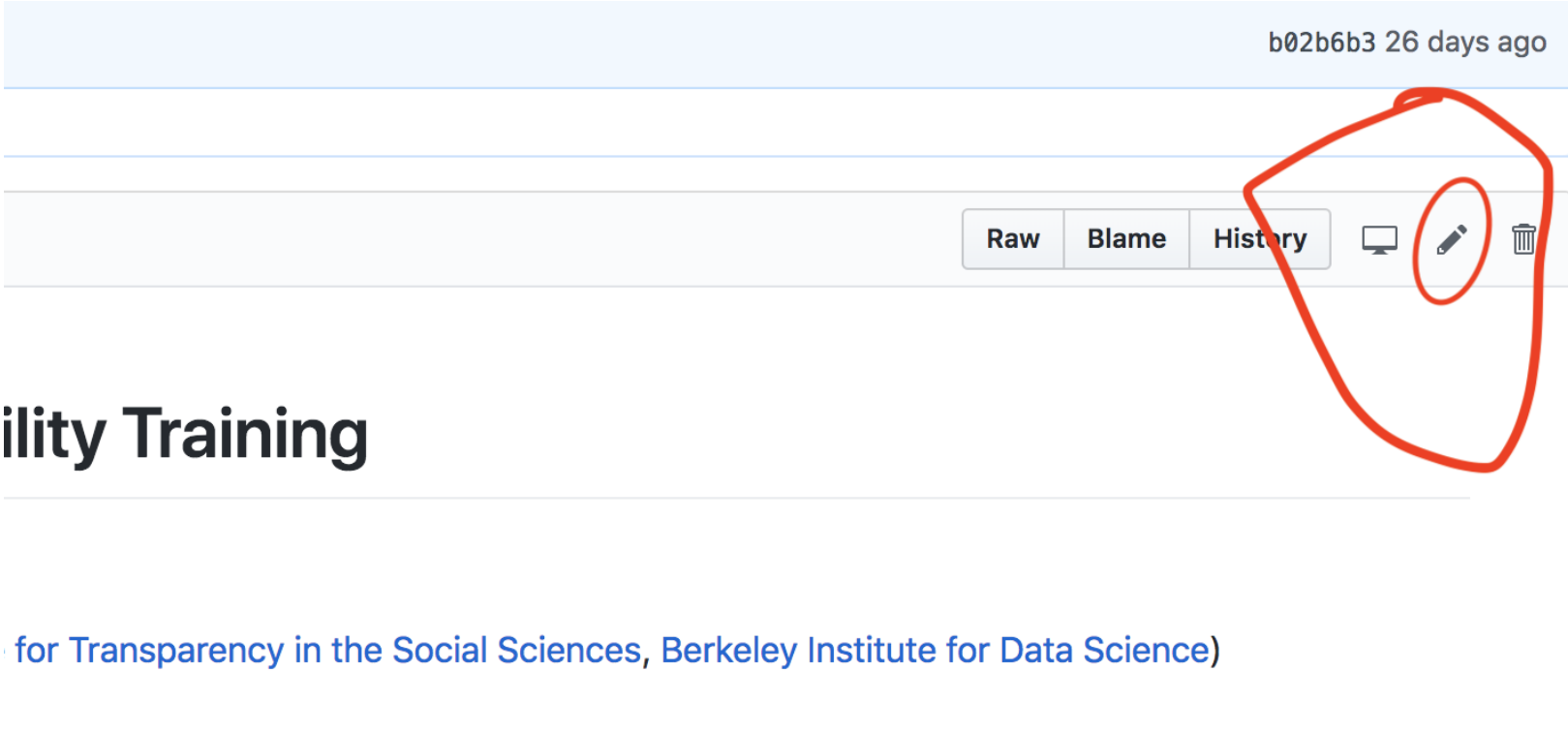
Cons: Harder adoption.

Tips to ease the adoption: concepts.

- **Git** is the software that does all the magic. **Github** is an implementation of Git that is easier to use, provides free (public) cloud service, and tools for collaboration.
- A repository (**repo**) is a master folder that contains all your work.
- Whenever you take a snapshot of your *saved* work, you **commit**
- When you make changes to your files in your computer, you are working **locally**, whenever you make changes to the files in the cloud you are working **remotelly**.
- Whenever you want to update your *remote repo*, you **push**. If you want to update you *local repo*, you **pull**.
- When you copy a repo from another person into your online github account, you **fork** it.
- When you start a repo from github.com (just created or forked), and want to download it for the first time to your local computer, you **clone** it.

Tips to ease the adoption: 4 min tutorial.

1- Create github.com account and sign in. [1 min]
2- Search `BITSS IDB` 3- Fork it.[1min]
4- (You have left the BITSS account, and are now in your account)
5- Click on `README.md`, then click on the pencil icon [1min]



Tips to ease the adoption: 4 min tutorial.

6- Modify the title (`# IDB Reproducibility Training`) and click `Commit changes`
7- Go back to the root folder of the repo (`IDBMarch2018`), and click `Clone or download`. And download as a zip.

If you had Github Desktop (an app from the Github company) installed, you could clone the repo and play more.

But I just wanted to give you a quick intro to:
- Find code of interest, fork it, edit it, make your first commit, and download.

Want to learn more:

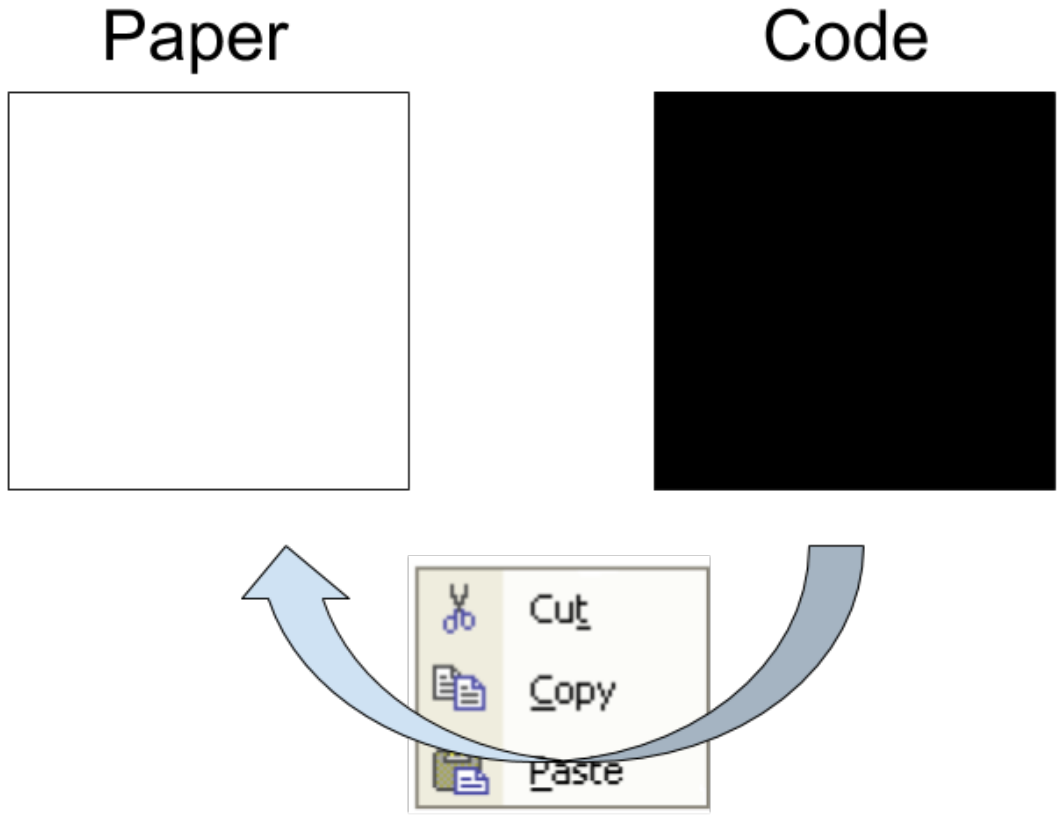
- Great 20 min intro to Git by Alice Bartlett
- Great 2hr tutorial to Github by Jenny Bryan (git ninja)
- Documentation from Matthew Gentzkow Jesse Shapiro
- Come to 3-Day trainning (RT2) in Seattle (Amsterdam next week, repo from last year)!

Dynamic Documents

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 (dyndocs release here and reviews here and here)

Currently code and narrative components live in separate universes



Dynamic Documents: integrate the two universes!

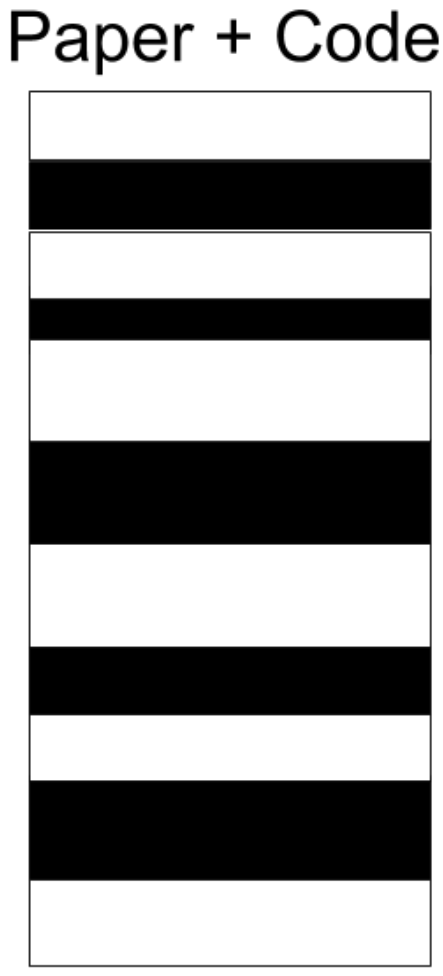


Figure 4:

Dynamic Documents: A Recipe

- 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

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)

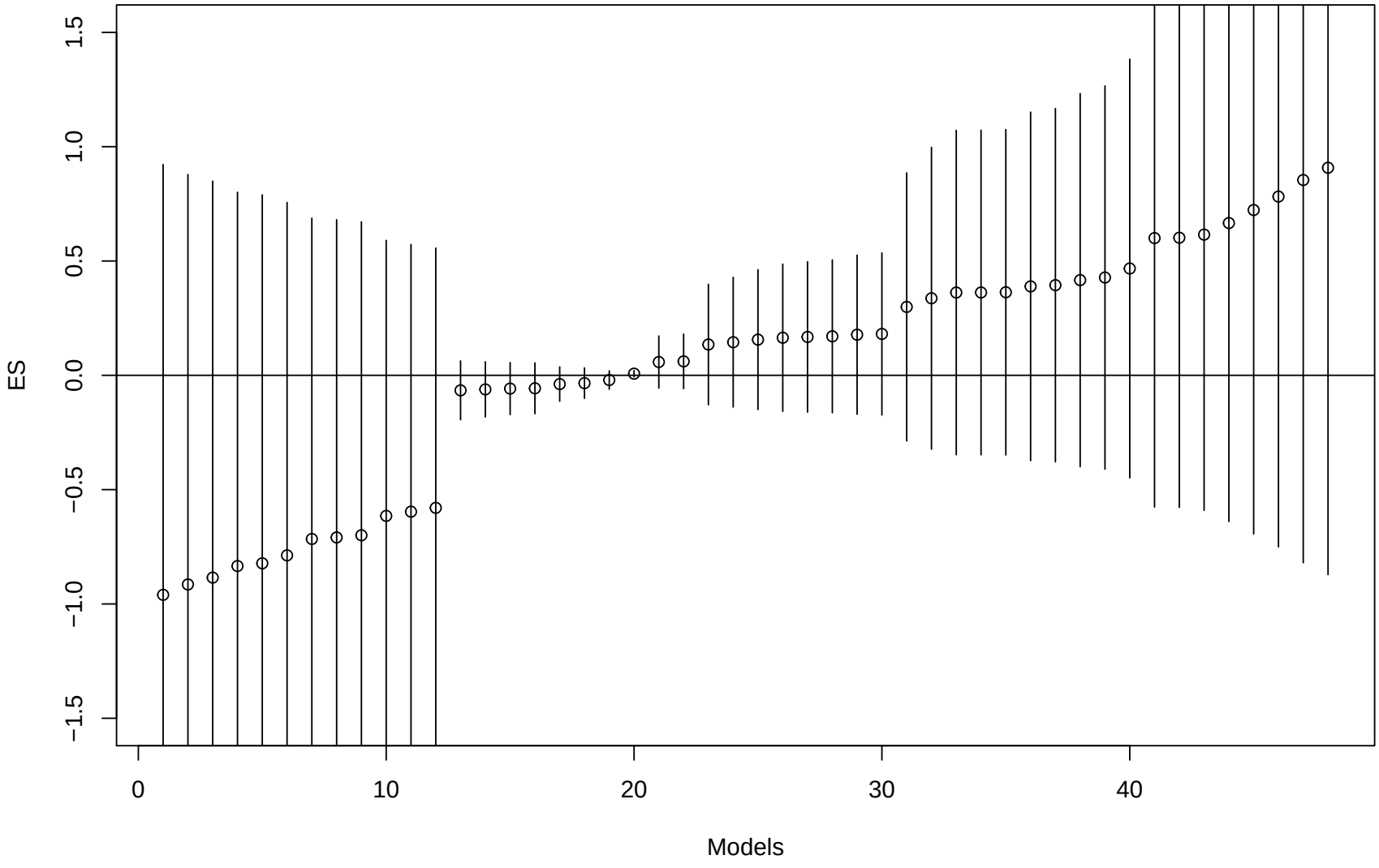
````
```{r, eval=TRUE}
here goes the code
```
````

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

P-hacking with the little experiment

- OLS
- 3 outputs
- 2 Treatment vars
- 7 Possible covariates (6 + none)
- Total of 42 plausible models

P-Hacking in Action (Specification Curve)



Back-up Demo: 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* $\frac{1}{365} \times N = 0.058$*?*

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:

$$
\begin{aligned}
1 - \bar{p}(n) &= 1 \times \left(1 - \frac{1}{365}\right) \times \left(1 - \frac{2}{365}\right) \times \cdots \times \left(1 - \frac{n-1}{365}\right) \\
&= \frac{365 \times 364 \times \cdots \times (365 - n + 1)}{365^n} \\
&= \frac{365!}{365^n (365 - n)!} = \frac{n! \cdot \binom{365}{n}}{365^n} \qquad (1) \\
p(n = 21) &= 0.444
\end{aligned}
$$

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 $n = 21$ 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 21 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 = 21
n.sims.param = 1e4
birthday.prob(n.pers.param,n.sims.param)
```

```
## [1] 0.4531
```

Results

- Many people originally think of a prob $\sim \frac{1}{365} \times N = 0.058$
- However the true probability is of $p(n = 21) = 0.444$
- And the simulated probability is of 0.4365

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?)

Bonus let's try and excercise in Stata (15)

1- Go to github.com and search `dyndoc tier` or click here: github.com/dvorakt/TIER_exercises.
2- Download or clone the repo.
3- Unzip it.
4- Open Stata (15) and type `dyndoc filepath/dyndoc_debt_growth/debt and growth stata dyndoc.do", replace` 5- Go to the folder and click in `debt and growth stata dyndoc.html`