

Dynamic Documents for Open Policy Analysis

Fernando Hoces de la Guardia
BITSS

-

Slides at
<https://github.com/BITSS/AIR2018>

American Institutes for Research, May 2018

Open Policy Analysis

Dynamic Documents For Computational Reproducibility

One Type of Dynamic Document: R Markdown

Practical Exercise #1

Practical Exercise #2

Practical Exercise #3

Final Remarks & More Resources

Open Policy Analysis

Research Transparency/Open Science

Issues:

- ▶ Scientific misconduct
- ▶ Publication Bias
- ▶ Specification searching / P-Hacking
- ▶ Replications problems

Solutions:

- ▶ Ethical research
- ▶ Registrations
- ▶ PAPs
- ▶ Guidelines and Protocols

New Dimension to Increase Transparency and Reproducibility: Policy Analysis

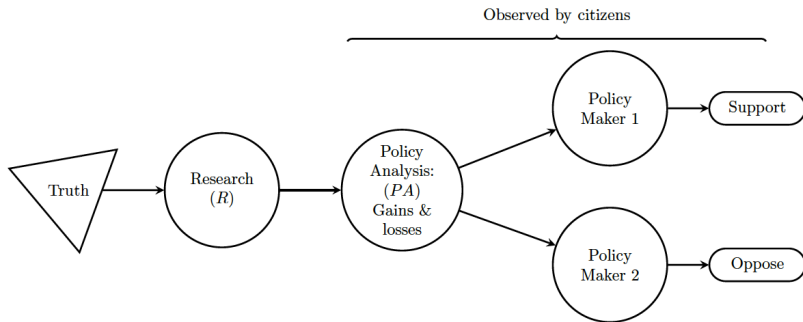


Figure 1: Policy-making with high credibility in research and policy analysis

Credibility Crisis

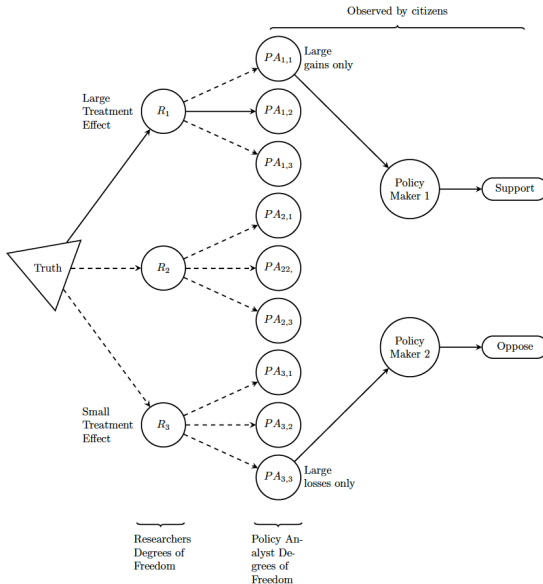
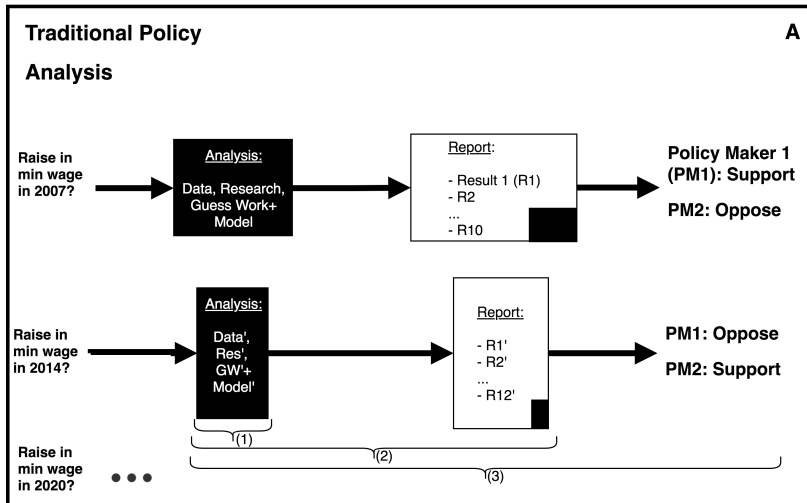


Figure 2: Policy-making with low credibility in research and policy analysis

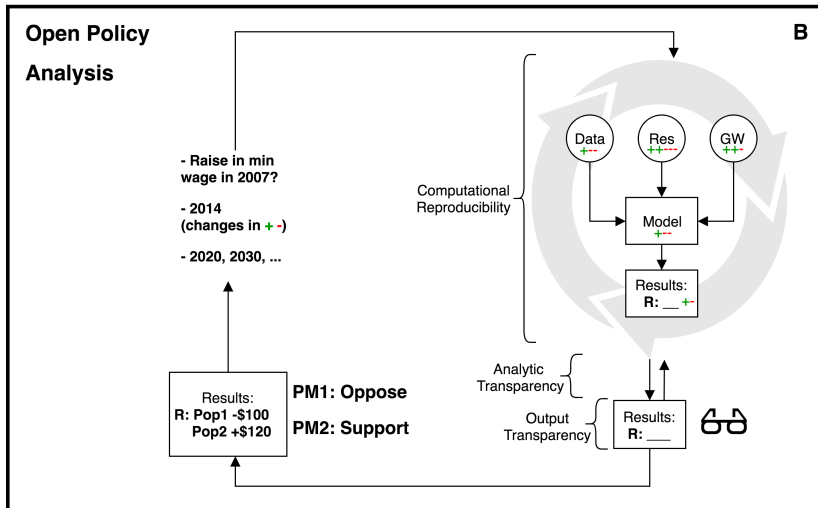
Our Proposal for Open Policy Analysis

- ▶ Increase awareness (Motivational paper [here.](#))
- ▶ Build guidelines and curriculum for open for policy analysis (similar to the TOP Guidelines for research).
- ▶ Partner with agencies/think tanks interested in implementing these ideas. [Example here.](#)
- ▶ Iterate.

Traditional Policy Analysis



Open Policy Analysis

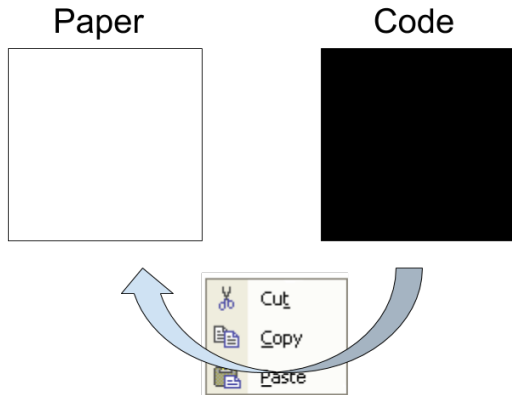


Dynamic Documents For Computational Reproducibility

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!

Paper + Code



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

Markdown language/syntax in 60 seconds

syntax

Plain text
End a line with two spaces to start a new paragraph.
italics and `_italics_`
****bold**** and `__bold__`
superscript²
~~~~strikethrough~~~~  
[\[link\]\(www.rstudio.com\)](#)

# Header 1  
## Header 2  
### Header 3  
#### Header 4  
##### Header 5  
##### Header 6

endash: --  
emdash: ---  
ellipsis: ...  
inline equation:  $A = \pi r^2$   
image:   
horizontal rule (or slide break):

## becomes

Plain text  
End a line with two spaces to start a new paragraph  
*italics* and *italics*  
**bold** and **bold**  
superscript<sup>2</sup>  
~~strikethrough~~  
[link](#)

## Header 1

## Header 2

## Header 3

### Header 4

### Header 5

### Header 6

endash: --

emdash: ---

ellipsis: ...

inline equation:  $A = \pi * r^2$

image:



One Type of Dynamic Document: R Markdown



## For our exercise: R Markdown

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



# R Markdown



# 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  
```
```

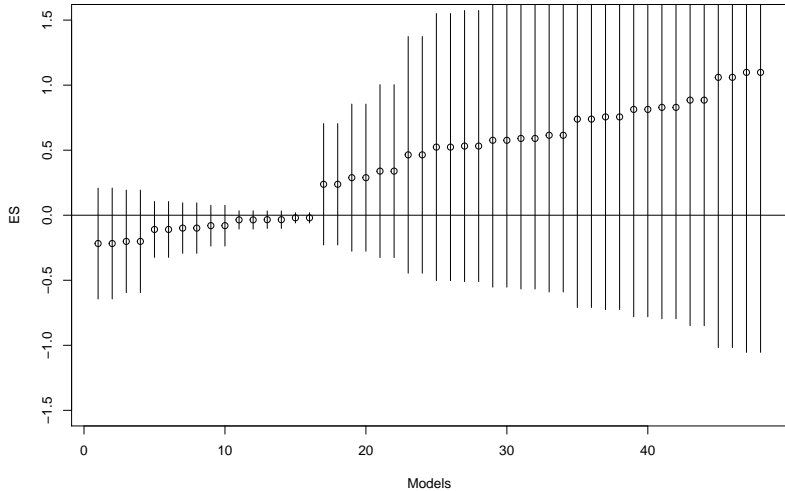
To write inline use only one Back-tick to open followed by an “r” and one to close ``r 1+1`` in the output.

# Little Demo: Our Little Experiment

## Can we p-hack it?

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

# P-Hacking in Action (Specification Curve)





## Practical Exercise #1

## Hands-on exercise: the birthday problem!

As an illustration let's write a report using the participants in this workshop to illustrate the famous [birthday problem](#).

*What is the probability that at least two people in this room share the same birthday?*

*Is it something like  $\frac{1}{365} \times N = 0.085$ ?*

# Goals for exercise #1

## **Primary Goals:**

- 1 - Become familiar with your first DD.
- 2 - Compile an empty (or default) DD into multiple formats.
- 3 - Edit a DD with some narrative, some code (in R) and some math (optional).
- 4 - Present all the results dynamically into multiple outputs.

# Goals for exercise #1

## **Primary Goals:**

- 1 - Become familiar with your first DD.
- 2 - Compile an empty (or default) DD into multiple formats.
- 3 - Edit a DD with some narrative, some code (in R) and some math (optional).
- 4 - Present all the results dynamically into multiple outputs.

## **Secondary Goal:**

- 1 - Expose you to some R programming.
- 2 - Entertain you with a fun problem.

## Create a new RMarkdown File

- 1 - In RStudio: File-> New File -> RMarkdown...
- 2 - Name it, and save it as /3-dynamicdocs/first\_dd.Rmd.
- 3 - Review/edit the header, and delete all the default body of text except for one code chunk.
- 4 - In that chunk define a seed (`set.seed(1234)`) and number of people in the room (`n.pers = ?`).
- 5 - Below the first chunk, write down a title (using #) and a brief description.

## The birthday problem: the math

Actually the math says otherwise:

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

$$p(n = 31) = 0.73$$

## Code for the math

(/3-dynamicdocs/first\_dd\_solution.Rmd)

Not relevant to look at: just copy and paste lines 23-30 from the solutions into your dynamic document.

```
\begin{align}
1 - \bar{p}(n) &= 1 \times \left(1 - \frac{1}{365}\right) \\
&\times \dots
\end{align}
```

A lot of equations using LaTeX syntax!

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

- 1 - Simulate 10,000 rooms with  $n = 31$  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 31 unique birthdays, across 10,000 simulations, and report the complement.



## Code for the simulation (/first\_dd\_solution.Rmd)

```
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,  
                            function(x)  
                              length(unique(x)) )  
  # Indicator with 1 if all are unique birthdays  
  all.different = 1 * (unique.birthdays==n.pers)  
  # Compute average time all have different birthdays  
  result = 1 - mean(all.different)  
  return(result)  
}  
  
n.pers.param = 31; n.sims.param = 1e4  
birthday.prob(n.pers.param,n.sims.param)
```

# Results

- ▶ Many people originally think of a prob  $\sim \frac{1}{365} \times N = 0.085$
- ▶ However the true probability is of  $p(n = 31) = 0.73$
- ▶ And the simulated probability is of 0.7293

## Practical Exercise #2

## Hands-on exercise #2: Mostly Harmless Econometrics!

There is a [fantastic Github](#) repo that is reproducing results from MHE

Lets use the of examples Figure [5.2.4](#) to show how dynamic docs can be used in data analysis.

# Figure to reproduce

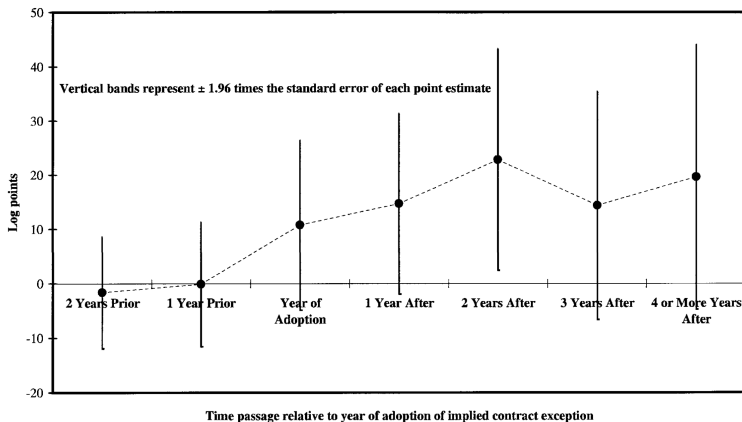


FIG. 3.—Estimated impact of implied contract exception on log state temporary help supply industry employment for years before, during, and after adoption, 1979–95.

## Goals for exercise #2

### **Primary Goals:**

- 1 - Demonstrate how the **entire workflow** of a study can fit into a DD.
- 2 - Show how to add options to the header.
- 3 - Demonstrate how a DD make code readable to non-coders.

## Goals for exercise #2

### **Primary Goals:**

- 1 - Demonstrate how the **entire workflow** of a study can fit into a DD.
- 2 - Show how to add options to the header.
- 3 - Demonstrate how a DD make code readable to non-coders.

### **Secondary Goal:**

- 1 - Expose you to some R programming.

## Instructions to get started with exercise #2:

- 1 - Create a new blank .Rmd file (steps 1 - 3 in from previous ex.)
- 2 - Save it as /3-dynamicdocs/Figure 5-2-4.Rmd
- 3 - Look at [this code](#) behind figure 5.2.4.
- 4 - Start building your own DD to describe what this code does.

We will go step by step using /3-dynamicdocs/Figure 5-2-4\_solutions.Rmd as back-up.



# Description

- ▶ Begin a new section (##), titled “Description”
- ▶ Write a brief description of our goal in the DD.
- ▶ You might want to insert a reference to the paper: [link here](#).
- ▶ Specific content not so relevant, just refer to “a treatment” and “a outcome”.

## Getting the raw data

- ▶ Begin a new section (##), titled “Raw Data”.
- ▶ Describe what you will do.
- ▶ Create two code chunks:

```
```{r download data, eval=FALSE, echo=TRUE,  
warning=FALSE, results='hide', message=FALSE}  
here goes the code  
```
```

# Cleaning the data

- ▶ Begin a new section (##), titled “Data Cleaning”.
- ▶ Describe what you will do:
  - ▶ Restrict sample to years between 1979 and 1995 (inclusive)
  - ▶ Guam from the sample (state = 98).
- ▶ Create one code chunk:

```
```{r data cleaning, echo=TRUE}  
here goes the code  
```
```

- ▶ Add some description on the data (using dynamic reporting).  
See solutions (Figure 5-2-4\_solutions.Rmd line 58) for examples.

# Build the analytic file

- ▶ Begin a new section (##), titled “Build analytic file”.
- ▶ Describe what you will do.
- ▶ We need to construct the following variables:
  - ▶ Log of total employment
  - ▶ Normalize the year variable to 1978
- ▶ Insert a new code chunk:

```
```{r analytic file, echo=TRUE}  
here goes the code  
```
```

## Describe the model to estimate (optional)

- ▶ Begin a new section (##), titled “Define model to estimate”.
- ▶ One line describing what we want to estimate (i.e. “We want to estimate a fixed effect model with lead and lag treatment variables”).
- ▶ A mathematical model that represents the equation to be estimated (look at solutions).

## Vizualize the results (optional)

- ▶ Begin a new section (##), titled “Vizualize the results”.
- ▶ One line describing what we want to estimate (i.e. “This estimates are then used to create figure 3 of the original paper, which is figure 5.2.4 in MHE.”).

```
```{r viz}  
here goes the code  
```
```

## Practical Exercise #2

- ▶ Run your version into multiple outputs.
- ▶ Run the solutions version into multiple outputs.
- ▶ Compare document with original version of the code.

## Practical Exercise #3



## Goals for exercise #3

### **Primary Goals:**

- 1 - Map the concepts of DD into Stata dyndoc.
- 2 - Demonstrate how to execute a DD in Stata.

## Hands-on exercise #3: Stata and TIER

- 1- Go to [github.com](https://github.com/dvorakt/TIER_exercises) and search dyndoc tier or click here: [github.com/dvorakt/TIER\\_exercises](https://github.com/dvorakt/TIER_exercises).
- 2- Download or clone the repo.
- 3- Unzip it.
- 4- Open Stata (15), set working directory, 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`

## Final Remarks & More Resources

## Final Remarks & More Resources

- ▶ With DD we can achieve a one-click reproducible workflow.
- ▶ This is particularly helpful to understand/present results that are hard to digest.
- ▶ More great examples in the workshop repo (4-moredynamicdocs).
- ▶ Want to learn more: [great free books](#) (can you guess how they were written?)