# Dynamic Documents

Aleksandr Michuda

Center for Data Science for Enterprise and Society, Cornell University

# Before We Begin

- Take-home Exam
- We will start presentation soon!

# Why Dynamic Documents?

Dynamic Documents are a part of the bigger picture of Reproducible Science. Sure, there is a fixed cost; **BUT**, they make my life easier in these ways:

- Short term: Easier to document fresh out of the oven results
- Medium term: Fast, reliable and tractable new results
- Long term: You can see how everything was created

# What are Dynamic Documents?

Based on principles of literate programming, we aim at combining code and paper in one single document

- Best framework to achieve the holy grail of one-click reproducible workflow
- Best implementations: Quarto.

# The State of Things Now

Currently, the code and the narrative components live in separate universes

# Part of Larger Workflow

- Dynamic documents are best used as part of a larger organized workflow
  - Structuring folders: Data, analysis, output
  - Documenting code
  - Combining both into a final document: Pre analysis or final paper
- We have already been using jupyter notebooks, but what if you want to turn that notebook into a publishable format for sharing or even submission?

# Using Markdown

- In terms of writing the "paper"/documentation part of dynamic documents, there are many solutions
  - Latex, HTML, RST (ReStructured Text)
- But most have honed in on using Markdown
  - Markdown is an easy way to write formatted text in a plain text format
  - But without as verbose and difficult of a syntax like latex/HTML
- Although basic markdown has the basics for formatting, creating tables, adding figures
- We will use Pandoc, which is used in both the Stata and R sessions

# What is Pandoc?

- Pandoc is sort of what it says: pan (all), doc (document)
- It's a way to convert between and across different file formats
  - Word -> HTML
  - Latex -> Markdown
  - HTML -> XML
  - Anything to anything
- See Pandoc's website for all input and output filetypes

# The Magic of Pandoc

- Pandoc and Markdown allows you to create one file that can then be used in many different places
- Example:
  - You're writing your CV and want to put it up in various places.
  - Your website needs HTML
  - One job posting allows PDF
  - One job posting only allows Word
- Ordinarily, you would need to have three versions, Word, HTML, PDF
  - This might get unruly as you change one but forget to change the other
  - What if there's another file format you might need?
- With Pandoc and markdown, you would:
  - write your CV in markdown
  - convert to PDF, Word and PDF with pandoc

# Quarto

- Quarto is the successor or RMarkdown, a very powerful dynamic document software developed for RStudio
- Now it also applies to jupyter notebooks and allows for creating nice documents from the notebooks you create!
- Dynamic documents usually have a `yaml` header at the top, that defines global options
  - We will talk about three special types of yaml options in Quarto:
    - output type
    - hiding code or output
    - execution options

# What is YAML?

- YAML is a very basic language created by Ansible (I think?) to define options for its software in a way that didn't require lots of coding.

```
---
title: "Toward a Unified Theory of High-Energy Metaphysics:
date: 2008-02-29
author:
  - name: Josiah Carberry
    id: jc
    orcid: 0000-0002-1825-0097
    email: josiah@psychoceramics.org
    affiliation:
      - name: Brown University
        city: Providence
        state: RI
        url: www.brown.edu
abstract: >
  The characteristic theme of the works of Stone is
```

# YAML

- But the basic thing you need in order to get the paper is:

```
---
title: My paper
author: Aleksandr Michuda
---
```

# Preview of what we will talk about

- Quarto is VERY expansive, you can write websites, books or dissertations with it
  - All with jupyter notebooks
- Today, we will focus on the basics that you will need if you wanted to write paper using a jupyter notebooks:
  1. Tables
  2. Figures
  3. Cross-references
  4. Citations

# Tables

- You can create tables easily in three ways:
  - Create your own markdown table (Not dynamic)
  - Put in a latex table directly (Not dynamic)
  - Generate a table from code (Dynamic)

Table 1: Demonstration of pipe table syntax

| Default | Left | Right | Center |
|---|---|---|---|
| 12 | 12 | 12 | 12 |
| 123 | 123 | 123 | 123 |
| 1 | 1 | 1 | 1 |

Table 2: copy-pasted table

| | A | B | C | D |
|---|---|---|---|---|
| 0 | 1.764052 | 0.400157 | 0.978738 | 2.240893 |
| 1 | 1.867558 | -0.977278 | 0.950088 | -0.151357 |
| 2 | -0.103219 | 0.410599 | 0.144044 | 1.454274 |

# Figures

- You can add figures that are generated or from a folder
- It's like figures markdown but with more options

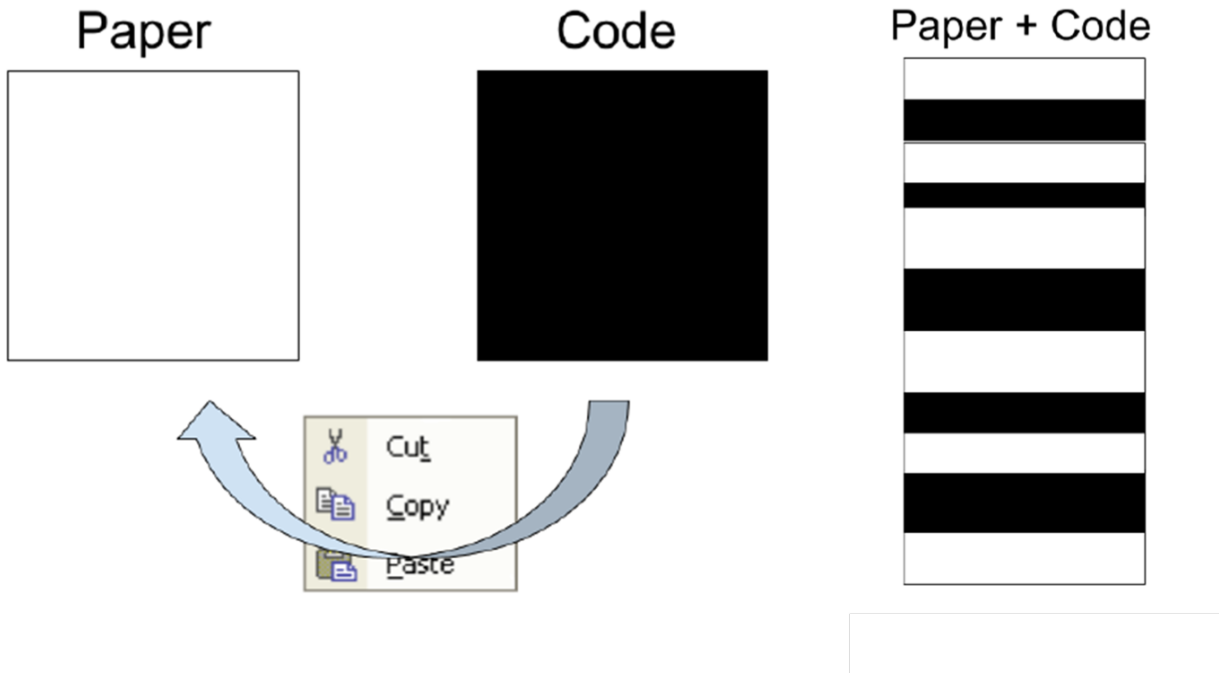


Figure 1: My figure 1, width 300

# Cross References

- To reference a figure, table or section, just use its label!
- See Table 3
- See Figure 4
- See Section **??**

# Citations and Footnotes

- For citations, You can use a standard bibtex file, just specify it in the YAML

```
---
bibliography: references.bib
---
```

- See Gupta et al. (2021) [1]
- This is true (Gupta et al. 2021).

[1] This is footnote.

# References

Gupta, Anubhab, Heng Zhu, Miki Khanh Doan, Aleksandr Michuda, and Binoy Majumder. 2021. "Economic Impacts of the COVID- 19 Lockdown in a Remittance-Dependent Region." *American Journal of Agricultural Economics* 103 (2): 466–85.