

# Academic writing in Markdown

Workshop on Reproducible Research and Modern Data  
Analysis

---

Sergio Correia

17 December, 2019


Board of Governors of the Federal Reserve

# Why Markdown?

---

# Suppose we want to produce this...

## Alternatives

1. Word
2. LaTeX
3. Markdown 

Time	Title	Presenter
14h15 – 15h30	How To Make A Pie	V.Orozco, C.Bontemps
16h00 – 17h15	Programming Tips	Sergiy Radyakin
17h15 – 18h30	Version control with git	Luis Fonseca

# In LaTeX:

```
1  \subsection{Alternatives}\label{alternatives}}
2
3  \begin{enumerate}
4  \def\labelenumi{\arabic{enumi}.}
5  \tightlist
6  \item Word
7  \item LaTeX
8  \item Markdown
9    \includegraphics[width=\textwidth,height=0.08333in]{md-logo.png}
10 \end{enumerate}
11
12 \begin{longtable}[]@{}lll@{}
13 \toprule
14 Time & Title & Presenter\tabularnewline
15 \midrule
16 \endhead
17 14h15 -- 15h30 & How To Make A Pie & V.Orozco, C.Bontemps\tabularnewline
18 16h00 -- 17h15 & Programming Tips & Sergiy Radyakin\tabularnewline
19 17h15 -- 18h30 & Version control with git & Luis Fonseca\tabularnewline
20 \bottomrule
21 \end{longtable}
```

# In Markdown:

```
1  ## Alternatives
2
3  1. Word
4  2. LaTeX
5  3. Markdown { height=8px }
6
7  | Time          | Title                      | Presenter          |
8  |-----|-----|-----|
9  | 14h15 - 15h30 | How To Make A Pie        | V.Orozco, C.Bontemps |
10 | 16h00 - 17h15 | Programming Tips         | Sergiy Radyakin     |
11 | 17h15 - 18h30 | Version control with git | Luis Fonseca         |
```

`\item \LaTeX{}` makes it `\emph{hard}` to  
focus on the writing

# Writing programming in LaTeX



**Tal Yarkoni**

@talyarkoni



the more time I spend in LaTeX, the more convinced I become that the whole thing was designed as an elaborate practical joke. the joke being, "let's create the world's greatest typesetting system, but embed it in the worst possible markup language."

8:10 PM · Dec 3, 2019 · [Twitter Web App](#)

**15** Retweets   **218** Likes



worst possible markup language."



- What can we do? Let's get rid of as much markup as possible!
- Markup -> **markdown**

# The goal of Markdown

“ A Markdown-formatted document should be publishable as-is, as plain text, without looking like it's been marked up with tags or formatting instructions. ”

John Gruber

# Who uses markdown?

You might already know Markdown:

- Reddit, Github, Stack Exchange, OpenStreetMap, etc. all use it
- Users of R (`rmarkdown`), Stata (~~dyndoc~~ `markstat`), Julia

You can try it out online at :

- <https://dillinger.io/>
- <https://jbt.github.io/markdown-editor/>

# Markdown Essentials

---

```
# This is a level-1 title
```

```
## This is a level-2 title
```

```
### This is a level-3 title
```

```
This is just a paragraph
```

## Syntax - Text

We can add text in *\*italics\**, **\*\*bold\*\***,  
~~~~striked~~~~ and as ``code()``.

Paragraphs are split by empty lines,  
so we are still in para 1.

Now we are in para 2.

This is amazing for working with Git!

---

We can add text in *italics*, **bold**, ~~striked~~ and as `code()`. Paragraphs  
are split by empty lines, so we are still in para 1.

Now we are in para 2. This is amazing for working with Git!

## Syntax - Unordered Lists

- First item
  - A subitem
  - Another subitem
- Last item

- 
- First item
    - A subitem
    - Another subitem
  - Last item

## Syntax - Ordered Lists

1. First item
2. Second one
1. Third one
1. Fourth one

- 
1. First item
  2. Second one
  3. Third one
  4. Fourth one



## Syntax - Quoted Text

We can add quotes:

> This text is quoted.

>

> And so is this

---

*This text is quoted.*

*And so is this*

## Syntax - Code blocks

You can paste pieces of code with three backticks before and after:

```
```
```

```
use "dataset", clear  
summarize somevariable
```

```
```
```

---

```
use "dataset", clear  
summarize somevariable
```

## Syntax - Code blocks

You can also set the syntax and other options:

```
``` stata
use "dataset", clear
summarize somevariable
```
```

---

```
use "dataset", clear
summarize somevariable
```

## Syntax - Tables

```
| Tables | Are | Cool |  
|-----|-----|-----|  
| col 1 is | first | $1600 |  
| col 2 is | second | $12 |
```

Becomes:

| Tables   | Are    | Cool   |
|----------|--------|--------|
| col 1 is | first  | \$1600 |
| col 2 is | second | \$12   |

However, I rarely type tables, but use Stata/R/Python (or <https://www.tablesgenerator.com> )

See [this page](http://www.example.com/) for more info



See **this page** for more info

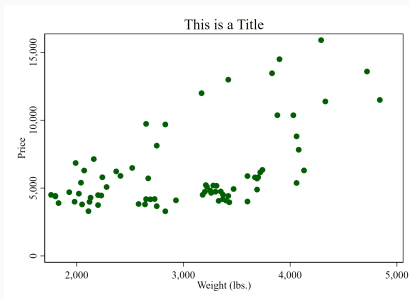
## Syntax - Figures

```

```

```
![Figure Caption](figure.png)
```

```
{width=50%}
```



## For more information

See these 3-10 minute online tutorials:

- <https://guides.github.com/features/mastering-markdown/>
- <https://www.markdownguide.org/cheat-sheet/>
- <https://commonmark.org/help/>
- <https://learnxinyminutes.com/docs/markdown/>

So far we've seen enough for a blog post, not for a paper.

What about ...

- Bibliographies and citations
- Document authors, title, abstract, etc.
- Including regression output (`esttab`, etc.) or arbitrary code

Enter pandoc.



# Pandoc

---

# Pandoc

---

From Wikipedia, the free encyclopedia

**Pandoc** is a [free and open-source document converter](#), widely used as a writing tool (especially by scholars)<sup>[1]</sup> and as a basis for publishing workflows.<sup>[2]</sup> It was created by [John MacFarlane](#), a philosophy professor at the [University of California, Berkeley](#).<sup>[3]</sup>

# What is pandoc?

- Pandoc is a “universal document converter”
- It can convert Markdown to LaTeX or to a PDF
- It can also convert Markdown to Word, PowerPoint, HTML, Jupyter, epub, and a *long* etc.

# How can pandoc help us?

- Headers at the beginning of a document
- Citations
- Extensible through “filters”

```
---  
author: John Smith  
title: An Interesting Paper  
abstract: Lorem ipsum...  
bibliography: something.bib  
---
```

```
# This is a title
```

```
And some text [@smith1776]
```

# How can pandoc help us?

```
> pandoc sample_article.md
  --filter=pandoc-citeproc
  --output=article.pdf
```

## An Interesting Paper

John Smith

### Abstract

Lorem ipsum...

### This is a title

And some text (Smith 1776)

### References

Smith, Adam. 1776. *An Inquiry into the Nature and Causes of the Wealth of Nations*. McMaster University Archive for the History of Economic Thought. <https://EconPapers.repec.org/RePEc:hay:hetboo:smith1776>.

# How can pandoc help us?

```
> pandoc sample_article.md  
  --filter pandoc-citeproc  
  --output=article.docx
```

## An Interesting Paper

John Smith

Lorem ipsum...

### This is a title

And some text (Smith 1776)

### References

Smith, Adam. 1776. *An Inquiry into the Nature and Causes of the Wealth of Nations*.  
McMaster University Archive for the History of Economic Thought. |

## Pandoc powers many tools

- R Markdown
- Stata's `markstat`
- Panwriter (<https://panwriter.com>) on Mac, Windows, Linux
- Typora (<https://typora.io>) on Mac, Windows, Linux
- Etc.

# The limits of pandoc

Working with pandoc is amazing, except perhaps for two things:

1. Pandoc is a command line tool, if you use it directly (instead of through the tools listed above) the command line gets *quite* long.
2. We often want to do things beyond what pandoc is capable.



## Pandoc and the command line (1/2)

This is the command line of this presentation, so far:

```
> pandoc --from=markdown --to=beamer --standalone  
  --pdf-engine=xelatex --output=markdown-correia-2019.pdf  
  --filter=panflute slides.md
```

It gets way longer for papers. You need to remember it every time.

## Pandoc and the command line (2/2)

Solutions:

1. Use a tool that takes you away from the command line
2. Use a wrapper, like `pandocomatic` (Ruby), `panrun` (in Python), etc.

None did *exactly* what I wanted, so I wrote **pandocmk**, to help me write papers and presentations. New line is just:

```
> pandocmk slides.md --view --watch
```

## Extending the abilities of pandoc: raw tex

- Pandoc allows you to write raw tex code (*if* your output is PDF)
- Suppose you want to include regression results:

In Stata:

```
esttab using "table1.tex", ...
```

In Markdown:

```
\input{"table1.tex"}
```

## Extending the abilities of pandoc: raw tex

- Output:

|          | (1)                | (2)                | (3)                |
|----------|--------------------|--------------------|--------------------|
| weight   | 4.903***<br>(6.63) | 5.703***<br>(5.17) | 5.055***<br>(5.04) |
| length   |                    | -40.11<br>(-0.98)  |                    |
| Turn FE  | Yes                | Yes                | Yes                |
| Trunk FE | No                 | No                 | Yes                |
| N        | 74                 | 74                 | 74                 |
| R2       | 0.593              | 0.600              | 0.635              |

*t* statistics in parentheses

\*  $p < 0.05$ , \*\*  $p < 0.01$ , \*\*\*  $p < 0.001$

## Extending the abilities of pandoc: filters

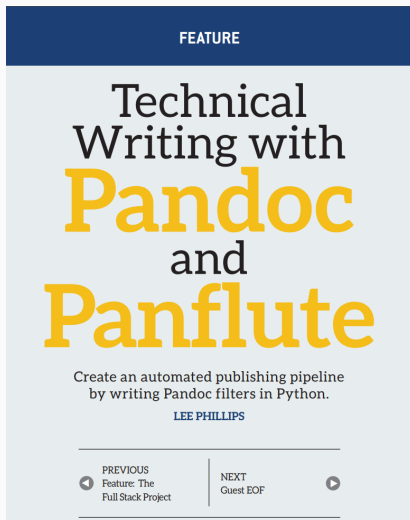
- Internally, pandoc represents all documents as objects made of elements (paragraphs, tables, headers, etc.).
- Advanced users can modify this internal representation through programs called “filters”.
- Although you could write filters in any programming language, my (biased) recommendation is to use Panflute (Python).

# Panflute

---

# Panflute

Has become popular for writing technical documents (Linux Journal, Feature Article):



... surprisingly popular:

## panflute

---

[PyPI page](#)

[Home page](#)

Author: Sergio Correia

License: BSD3

Summary: Pythonic Pandoc filters

Latest version: 1.12.4

Downloads last day: 188

Downloads last week: 1,311

Downloads last month: 4,775



# What exactly is a panflute filter?

This Python function changes all italics to bold:

```
1     ##### myfilter.py #####
2     def myfilter(element, document):
3         if type(element) == panflute.Emph:
4             return panflute.Strong(*elem.content)
5
6     panflute.run_filter(mytype, document)
7     #####
```

Panflute runs the function for all elements within a document, and returns a modified document:

```
> pandoc mydoc.md --filter=myfilter.py --output=mydoc.pdf
```

- Advanced enough users will eventually face something requiring a filter.
- I have used filters to:
  - Move the appendix section to the end (after the references).
  - Build tables from CSV files.
  - Run Stata/Python/R and show the results.
  - Build automated reports based on online data.
  - Etc.

# Summary

- Markdown makes writing easier compared to LaTeX, while preserving the good-quality output
- Advanced users can use pandoc to extend it as needed (plus **pandocmk**, **panflute**, etc.)
- All in all, it plays well with other tools for reproducible research:
  - Easy to track with version control (Git)
  - Can be used from/to statistical software (Stata, R)



## Useful references

- Download Pandoc: <https://pandoc.org/installing.html>
- Download Python: <https://www.python.org/downloads/>
- Download Panflute:  
<https://github.com/sergiocorreia/panflute#install>
- Create markdown tables online:  
[https://www.tablesgenerator.com/markdown\\_tables](https://www.tablesgenerator.com/markdown_tables)

# Panflute Example

Suppose you have this markdown file:

```
----- mydocument.md -----  
Let's write a calculator in markdown:  
  
``` calculator  
16 + 13 * 2  
```  
  
-----
```

And want all code blocks of the `calculator` type to compute and show the answer.

# Panflute Example

Then this short Python filter would be all you need:

```
1 ##### calculator.py #####
2 import panflute as pf
3
4 def action(e, doc):
5     if isinstance(e, pf.CodeBlock) and 'calculator' in e.classes:
6         val = eval(e.text)
7         e.text = f'{e.text} = {val}'
8
9 if __name__ == '__main__':
10     pf.run_filter(action)
```

# Panflute Example

Command line:

```
> pandoc mydocument.md  
    --filter=calculator.py  
    --output=mydocument.pdf
```

Resulting PDF:

Let's write a calculator in markdown:

```
16 + 13 * 2 = 42
```