

# Publication-ready data analysis with Stata

E. F. Haghish

Department of Psychology, University of Oslo



## Material are hosted on GitHub:

- ▶ repository: <https://github.com/haghigh/promenta>
- ▶ an example of and **organized project** for running a simple CFA analysis and producing a dynamic document
- ▶ The current slides
- ▶ markdown template for producing PDF slides within Stata
- ▶ Rmarkdown template for creating PDF slides within RStudio
- ▶ **markdoc** package  
<https://github.com/haghigh/markdoc>
- ▶ **rcall** package <https://github.com/haghigh/rcall>
- ▶ PDF of journal articles relevant to this lecture



Figure 1: Relevant literature

# Overview

## Part 1

- ▶ Reproducible Research
  - ▶ Challenges of reproducible researcher
  - ▶ Challenges of teaching statistics in all fields of science
  - ▶ How can we improve research reproducibility?

## Part 2

- ▶ Automate the process of data analysis
  - ▶ organizing a computational project
  - ▶ reproduce the *entire* analysis
  - ▶ Integrating version control (PDF article is provided)
  - ▶ Automate multiple statistical software
  - ▶ Interfacing R into Stata for data analysis
  - ▶ A brief introduction to `rca11` package

# Overview

## Part 3

- ▶ Automatize the process of reporting
  - ▶ producing a sensible analysis reports for a manuscript
  - ▶ dynamic tables, dynamic graphs, and dynamic text
  - ▶ discuss its necessity
- ▶ markdoc software
  - ▶ Installing markdoc from GitHub
  - ▶ workflow
  - ▶ Markdown notation
  - ▶ markdoc commands and its markup notations
- ▶ Using `markdoc` in classroom
  - ▶ presentation slides within Stata
- ▶ Using `markdoc` for software documentation
  - ▶ Stata help files
  - ▶ Package vignette or supplementary web material

# Part 1: Reproducible Research

# Statement of problem

Garfield (1995) defines learning statistics as follows:

1. learning to communicate using statistical language
2. solving statistical problems
3. drawing conclusions
4. supporting conclusions with statistical reasoning

## statistical education requires:

- ▶ in-depth understanding of statistical concepts
- ▶ statistical reasoning
- ▶ computer programming skills

## Anxiety among graduate students

- ▶ Statistics generally causes inconvenience for researchers of different fields (Baloglu, 2003)
- ▶ 80% of graduate students suffer from statistics anxiety (Onwuegbuzie, 2004)
  - ▶ math anxiety
  - ▶ computer anxiety
  - ▶ programming anxiety



- ▶ Proper statistical education has been *avoided*
  - ▶ Teaching through GUI instead of programming
- ▶ The complexity of the methods is increasing annually
  - ▶ The journals' appetite for intricate statistics is growing
- ▶ The gap between statistical education and statistical practice is increasing
- ▶ There is no statistical software that does **everything**
  - ▶ We might need to learn several statistics software
  - ▶ Particular analyses might be available in a special software

# Problem?

We are lacking

- ▶ Basic coding education (no more mouse-and-click)
- ▶ Skills for planing and organizing data analysis
- ▶ Tracking our potential errors in different steps of research
- ▶ Communicating statistical decisions and reasons

Which results in lacking **reproducibility**

# Why Most Published Research Findings Are False

John P. A. Ioannidis

## Summary

There is increasing concern that most current published research findings are false. The probability that a research claim is true may depend on study power and bias, the number of other studies on the same question, and, importantly, the ratio of true to no relationships among the relationships probed in each scientific field. In this framework, a research finding is less likely to be true when the studies conducted in a field are smaller; when effect sizes are smaller; when there is a greater number and lesser preselection of tested relationships; where there is greater flexibility in designs, definitions, outcomes, and analytical modes; when there is greater financial and other interest and prejudice; and when more teams are involved in a scientific field in chase of statistical significance. Simulations show that for most study designs and settings, it is more likely for a research claim to be false than true. Moreover, for many current scientific fields, claimed research findings may often be simply accurate measures of the prevailing bias. In this essay, I discuss the implications of these problems for the conduct and interpretation of research.

factors that influence this problem and some corollaries thereof.

## Modeling the Framework for False Positive Findings

Several methodologists have pointed out [9–11] that the high rate of nonreplication (lack of confirmation) of research discoveries is a consequence of the convenient, yet ill-founded strategy of claiming conclusive research findings solely on the basis of a single study assessed by formal statistical significance, typically for a  $p$ -value less than 0.05. Research is not most appropriately represented and summarized by  $p$ -values, but, unfortunately, there is a widespread notion that medical research articles

## It can be proven that most claimed research findings are false.

should be interpreted based only on  $p$ -values. Research findings are defined here as any relationship reaching formal statistical significance, e.g., effective interventions, informative predictors, risk factors, or associations. “Negative” research is also very useful

is characteristic of the field and can vary a lot depending on whether the field targets highly likely relationships or searches for only one or a few true relationships among thousands and millions of hypotheses that may be postulated. Let us also consider, for computational simplicity, circumscribed fields where either there is only one true relationship (among many that can be hypothesized) or the power is similar to find any of the several existing true relationships. The pre-study probability of a relationship being true is  $R/(R + 1)$ . The probability of a study finding a true relationship reflects the power  $1 - \beta$  (one minus the Type II error rate). The probability of claiming a relationship when none truly exists reflects the Type I error rate,  $\alpha$ . Assuming that  $c$  relationships are being probed in the field, the expected values of the  $2 \times 2$  table are given in Table 1. After a research finding has been claimed based on achieving formal statistical significance, the post-study probability that it is true is the positive predictive value, PPV. The PPV is also the complementary probability of what Wacholder et al. have called the false positive report probability [10]. According to the 2

Figure 2: Why most published research findings are false

## RESEARCH ARTICLE

## PSYCHOLOGY

# Estimating the reproducibility of psychological science

Open Science Collaboration\*†

Reproducibility is a defining feature of science, but the extent to which it characterizes current research is unknown. We conducted replications of 100 experimental and correlational studies published in three psychology journals using high-powered designs and original materials when available. Replication effects were half the magnitude of original effects, representing a substantial decline. Ninety-seven percent of original studies had statistically significant results. Thirty-six percent of replications had statistically significant results; 47% of original effect sizes were in the 95% confidence interval of the replication effect size; 39% of effects were subjectively rated to have replicated the original result; and if no bias in original results is assumed, combining original and replication results left 68% with statistically significant effects. Correlational tests suggest that replication success was better predicted by the strength of original evidence than by characteristics of the original and replication teams.

Figure 3: Estimating the reproducibility of psychological science

# Reproducibility vs. Replication

- ▶ The two terms have been used interchangeably (Loscalzo, 2012)
- ▶ They have different meanings in different fields of science
- ▶ Replication requires re-implementing experiments by other research groups (Baggerly & Berry, 2009)
- ▶ using either the same methodology or alternatives
- ▶ Problems with replication?

# Reproducibility

- ▶ Baggerly & Berry (2009):
- ▶ reproducibility is replicating the computation by an independent researcher
  - ▶ using the same data, programmed code, procedure, and methodology
  - ▶ and without requiring any further assistance or information from the author (King, 1995)
- ▶ the least standard for evaluating the quantitative results
- ▶ reproducibility does not guarantee (Peng, 2011; Stodden, et. al., 2014):
  - ▶ quality or sound methodology
  - ▶ accurate data collection
  - ▶ validity of the findings
- ▶ reproducibility grants limited transparency (Gentleman & Lang, 2012)
  - ▶ validate the computational procedure
  - ▶ check or adapt the claims in the scientific publication

# Sources of error in research

- ▶ Errors can happen at any stage of research
  - ▶ study design
  - ▶ data collection
  - ▶ digitizing the data from questionnaires to a computer
  - ▶ cleaning the data
  - ▶ preparing the data for analysis
  - ▶ choice of methodology
  - ▶ adjustment options, analytical assumptions, algorithms, etc. . .
  - ▶ interpreting the results
  - ▶ reporting the results in the publication
  - ▶ copy-pasting from statistical software to MS Word
  - ▶ any problem with that?
  - ▶ . . .
- ▶ Or afterwards, such as publication bias, etc. . .

# Collaboration on computational research

- ▶ The majority of statistical contributions do not appear in the manuscripts
  - ▶ no code, no data checking, no quality assurance, ...
- ▶ Lacking reproducibility means no collaboration on statistical analyses
- ▶ Collaboration on statistical analysis is like collaboration on software:
  - ▶ well-structured
  - ▶ automatized
  - ▶ well-documented
  - ▶ dependencies are carefully planned, organized, documented



# Costs

- ▶ You need to learn new tricks and let go of old habits
- ▶ No one gives you credit for being transparent
- ▶ Transparency means your mistakes can be revealed by others
  - ▶ Shame or gratitude?
  - ▶ What you cannot reproduce your own analysis?
  - ▶ How would you feel about sharing your code?
- ▶ Reproducibility is human problem, not computers

## **Part 2: Automated Data Analysis**

# Automated Data Analysis

- ▶ Automated data analysis means making data analysis reproducible
  - ▶ writing analysis code to track **the entire data analysis**
  - ▶ setup and organize your analytic project
  - ▶ Operating system and statistical software
  - ▶ Add-on packages
  - ▶ Data management
  - ▶ Nesting analysis code (and why should you)
  - ▶ Communicating the analysis



Figure 4: The procedure we are intending to automatize

# Organizing the computation

- ▶ The rule is to be disciplined, **very disciplined**
  - ▶ Keep track of changes in code, data, and analysis results
- ▶ There is no template to be applicable to all projects
  - ▶ with different types of data, there will be different procedures and workflows
- ▶ Rule of thumb:
  - ▶ protect your raw data
  - ▶ keep track of all the code for preparing the data for analysis
  - ▶ keep track of all the analysis code
  - ▶ create separate directories for storing raw data, code, analysis results/reports, documents, etc.

## UiO: Example 1: R package

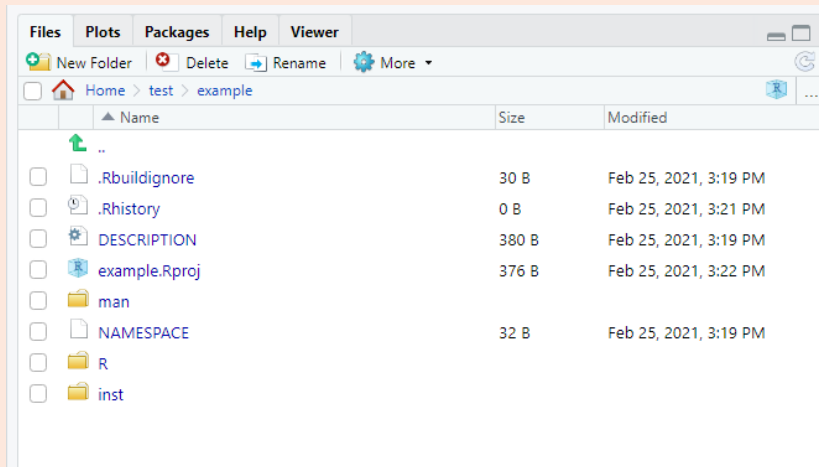


Figure 5: R Package Structure

## Example 2: UiN Project








Name	Date modified	Type	Size
 anadata	11.02.2021 08:44	File folder	
 clean	08.02.2021 12:28	File folder	
 code	22.02.2021 10:30	File folder	
 docs	08.02.2021 11:35	File folder	
 raw	12.02.2021 22:43	File folder	
 README	19.02.2021 09:44	Text Document	1 KB

Figure 6: Young in Norway Study

## Example 3: My personal preference



A screenshot of a file explorer window showing a project directory. The table lists files and folders with their names, modification dates, types, and sizes. The 'README' file is highlighted in blue.








Name	Date modified	Type	Size
 code	25.02.2021 15:30	File folder	
 data	22.02.2021 11:46	File folder	
 docs	25.02.2021 15:30	File folder	
 report	25.02.2021 15:30	File folder	
 results	25.02.2021 15:30	File folder	
 MAIN	22.02.2021 12:00	DO File	1 KB
 README	09.02.2021 13:23	Text Document	1 KB

Figure 7: My way of organizing a computational project



## Uio: Example of a README file

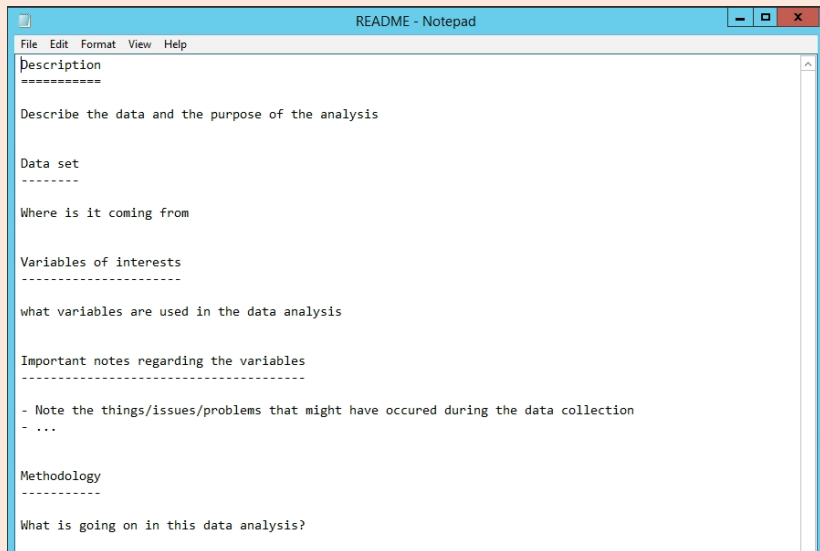


Figure 9: Writing a README file that is worth reading

# Nesting script files

- ▶ The idea is borrowed from computer science
  - ▶ For example, see how Linux kernel is compiled
  - ▶ see the **Makefile** in <https://github.com/torvalds/linux>
  - ▶ the file provides all of the orders to compile Linux from the source code
- ▶ We apply the same discipline to approach reproducibility
  - ▶ There will be a single file that provides the instructions to rerun the entire data analysis
  - ▶ I name that file **MAIN**, you name it . . .
  - ▶ the file will source all other script files used for preparing, analyzing, and reporting the analysis
- ▶ Nesting works best with relative file paths (instead of absolute paths)
  - ▶ begin the **MAIN** file by setting the working directory:
  - ▶ Use `setwd()` in **R**, `cd` in **Stata** and **SPSS**

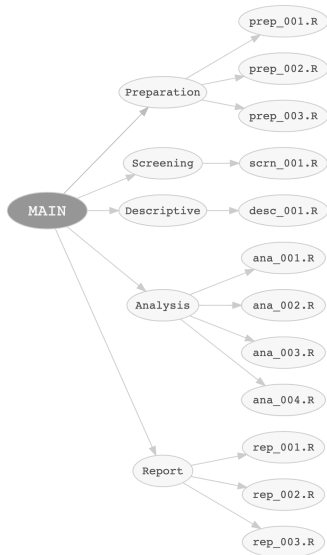


Figure 9: Nesting analysis files

# Nesting script files

## R

```
source('./code/preparation/prep_001.R')
```

## Stata

```
do './code/preparation/prep_001.do'
```

## SPSS

```
INSERT FILE='./code/preparation/prep_001.sps'.
```

# Notes: general suggestions

1. The raw data is kept untouched
  - ▶ Store time-consuming operations in a different directory (e.g. *anadata*)
2. Organize your code under subdirectories (if you write many files?)
3. Save the results (analysis outputs) in separate directories and name them properly within the code
4. Name and document your data file, especially if it is going to receive further updates in the future
5. Document the software dependencies (Operating system, R/Stata/etc. version, **ALL add-on packages'** versions)
  - ▶ check for example lavaan change history:  
<https://lavaan.ugent.be/history/>
6. Document the data set
  - ▶ use datadoc (Haghish, 2020) command for **Stata** or Rd documentation from **RStudio**

# Notes: data documentation

- ▶ CRAN recommends the following documentation section for a data set
- 1. Title, the label of the dataset, and where it was published (package name)
- 2. Description
- 3. Format, including a table summarizing the variables' types and labels
- 4. Notes attached to the dataset or the variables (for Stata only)
- 5. The source of the data; that is, where they are coming from
- 6. References, if any
- 7. Examples, if needed

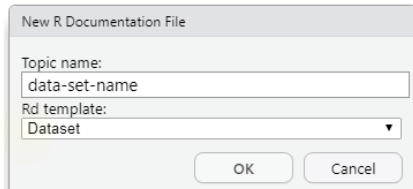
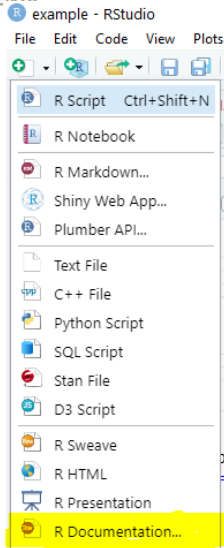


Figure 10: Data documentation in R

☰ README.md 

v. 1.0

# **datadoc : automatic data documentation**

## **help file generator**

---

### **Installation**

---

The `github package` is the only recommended way for installing `datadoc` . Once `github` is installed, type:

```
github install haghish/datadoc, stable
```

### **Description**

---

datadoc generate a documentation template for the data that is currently loaded in Stata

Figure 11: Data documentation in Stata



# Working with multiple statistical software

- ▶ With the speed of statistical development, sometimes we might need to use a particular method that is not available in Stata
- ▶ Typically, most graduate students implement statistical methods in R or Python
- ▶ Stata allows interfacing Python for data analysis or programming

*type `help python` for more information*

- ▶ The `rca11` package provide the same capabilities for interfacing R into Stata, and much more!
- ▶ Using such a strategy allows you to automate your analysis entirely within Stata and keep it reproducible

- ▶ **rcall** is hosted on GitHub only, not SSC

```
net install github, from("https://haghish.github.io/github/  
github install haghish/rcall, stable
```

- ▶ rcall automatically finds R on your machine. If it fails to do so, specify the path to R:
  - ▶ `rcall setpath "path/to/R"` will specify the path permanently
- ▶ rcall can automatically communicate dataset, variables, matrices, and scalars between R and Stata.
- ▶ For example, you can call R to run an analysis on the data loaded in your Stata and return the matrices and results to Stata automatically

# rcall example

## Passing a matrix from R to Stata

```
. rcall: A = matrix(1:6, nrow=2, byrow = TRUE)
```

```
. mat list r(A)
```

```
r(A)[2,3]
```

	c1	c2	c3
r1	1	2	3
r2	4	5	6

## Passing dataset from Stata to R

```
. sysuse auto, clear  
(1978 Automobile Data)  
  
. rcall: data <- st.data()  
  
. rcall: dim(data)  
[1] 74 12
```

# Part 3: Automated Analysis Reporting

# Avoiding manual reporting

I noted that errors can happen in the process of reporting

## Sources of error in research

- ▶ Errors are everyday and can happen at any stage of research
  - ▶ ...
  - ▶ interpreting the results
  - ▶ reporting the results in the publication
  - ▶ copy-paste from statistical software to MS Word
  - ▶ updating the report after making a change in the data or analysis
  - ▶ ...

# Avoiding manual reporting

- ▶ A solution is to do the data analysis and write the analysis report at the same time
- ▶ This is a paradigm borrowed from computer science, for solving software documentation problem
  - ▶ documentation is written within code files using special comment signs
  - ▶ next, a program extracts and renders the documentation and updates the documents (Knuth 1983)
- ▶ There are software for generating data analysis reports:
  - ▶ for **R**, use `rmarkdown` (Yihui, et. al., 2018)
  - ▶ for **Stata**, use `markdoc` (haghish, 2016)
  - ▶ both provide a restricted framework to examine the reproducibility of the code

# Literate programming

- ▶ The big problem of software documentation
- ▶ The literate programming solution
- ▶ Adaption of the literate programming in statistics
  - ▶ Should ideally supports real-time documentation
  - ▶ Should examine the analysis
  - ▶ Should provide a restricted framework to improve the code development



# markdown package

## Note about **markdown**

**MarkDoc** was developed for Stata in 2012. It comes in two versions, full-version (required additional third-party software) and mini-version (completely written within Stata).

*If you use **secure servers** or **restricted machines**, use the mini-version*

*The mini-version can be executed by adding `mini` option or by using the **mini** command.*

# markdown package

- ▶ markdown is a general purpose literate programming software
- ▶ developed particularly for Stata
- ▶ markdown is versatile:
  - ▶ generate publication-ready analysis report in various document formats (PDF, Docx, ODT, HTML, LaTeX, etc.) + includes a syntax highlighter
  - ▶ generate dynamic presentation slides
  - ▶ generate dynamic Stata help files in STHLP format or create a package vignette
- ▶ Analysis documentation/interpretation is written within *do-files*, as usual
- ▶ It emphasizes code readability by keeping the documentation simple

# MarkDoc features

- ▶ It works with the usual workflow of Stata do-files
  - ▶ It is easy to use
  - ▶ It underscores clean and readable documentation
- ▶ recognizes multiple markup languages
- ▶ has a built-in syntax highlighter
- ▶ supports several output documents
  - ▶ develops text documents
  - ▶ presentation slides
  - ▶ software documentation

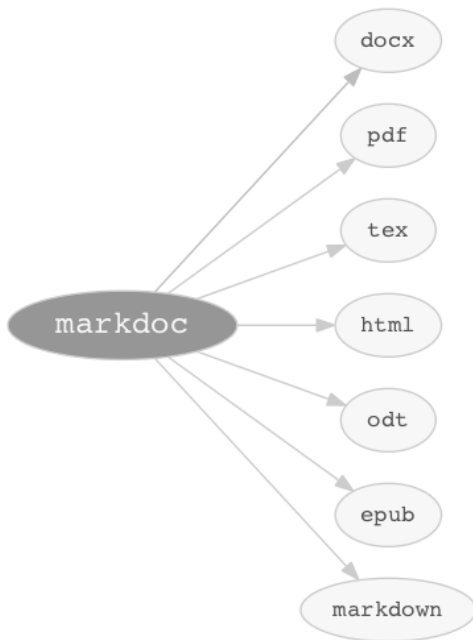


Figure 12: Supported document formats

# Dialog box

markdoc was designed to be a very user-friendly package. To further facilitate learning markdoc, a dialog box was programmed to visualize the main options and functionalities of the package.

- ▶ The dialog box includes three tabs, each specializes in a particular document format
  - ▶ dynamic document
  - ▶ presentation slide
  - ▶ package vignette

*To lunch the dialog box type:*

```
db markdoc
```

MarkDoc

Dynamic Document   Presentation Slide   Package Vignette

Select "smcl" or "do" file

Browse...

Markup language   Document format

markdown   • html   ○ pdf   ○ tex   ○ docx   ○ odt   ○ md

Options

☐ Install required software

☒ Replace the existing document

☒ Create HTML or LaTeX template

☒ Statax syntax highlighter

☐ Table of content

☐ Add the current date

☐ Count Stata commands

☐ Execute MarkDoc noisily

Select document layout style

stata

Use layout (css, docx, tex, etc)

Browse...

Title

Author

Affiliation

Address

Summary

? ®

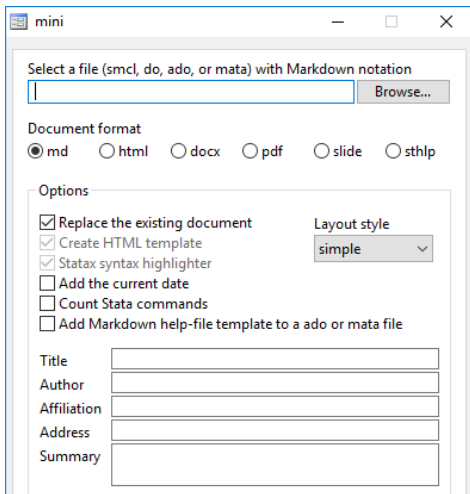
Submit   Cancel   OK

Figure 13: The dialog box

# Dialog box for markdoc 5.0

Use the `mini` dialog box, if third-party software are not installed.

```
. db mini
```



# Who can use `markdoc`?

`markdoc` was designed having learners in mind. It offers a GUI, a syntax highlighter, and plenty of features to encourage beginners to use it.

1. Students - as early as introductory statistics courses - can use `markdoc` to actively take note inside Stata Do-file Editor
2. University lecturers who teach statistics using Stata, can use `markdoc` to generate PDF slides, educational materials
3. Statisticians can use `markdoc` for creating dynamic analysis reports
4. Finally, advanced users and Stata programmers can use `markdoc` to generate dynamic help files and package vignettes



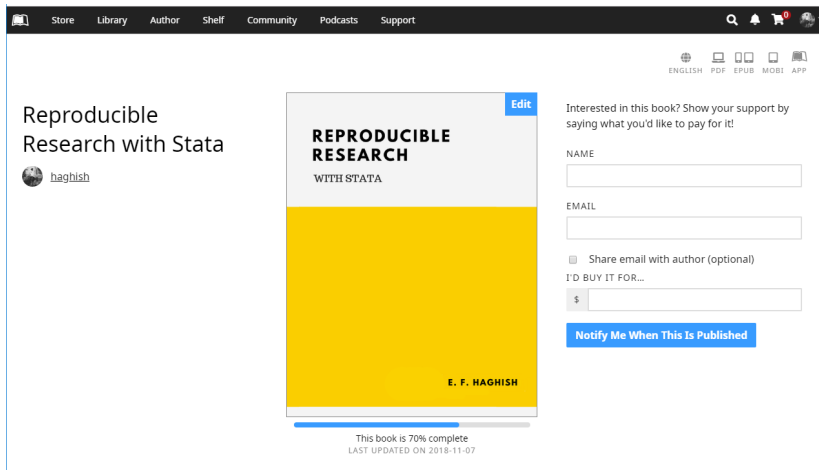


Figure 15: The presentation slides are based on a book

# markdoc Installation

- ▶ markdoc is hosted on GitHub only  
`https://github.com/haghigh/markdoc`
- ▶ markdoc has package dependencies which are:
  - ▶ weaver
  - ▶ datadoc
  - ▶ md2smcl
  - ▶ statax
- ▶ The `github` command can install `markdoc` and its dependencies. You can install the `github` command as follows:

```
. net install github, from("https://haghigh.github.io/github")
```

- ▶ Once the `github` command is installed, installing any Stata package from GitHub would be easy
- ▶ The installation only requires the authors' GitHub `username` and the `repository` name, separated by a slash
- ▶ For example, to install or update `markdoc` and its dependencies type the following command:

```
. github install haghish/markdoc, stable
```

Alternatively, you can use `github` command to search for `markdoc` package in GitHub by typing:

```
. github search markdoc
```

Repository	Username	Install	Description
<code>markdoc</code>	<code>haghigh</code>	<code>Install</code> <code>8610k</code>	A literate programming package for Stata which develops dynamic documents, slides, and help files in various formats homepage <a href="http://haghigh.com/markdoc">http://haghigh.com/markdoc</a> updated on 2018-10-15 <b>Hits:350 Stars:34 Lang:Stata</b> ( <a href="#">dependency</a> )

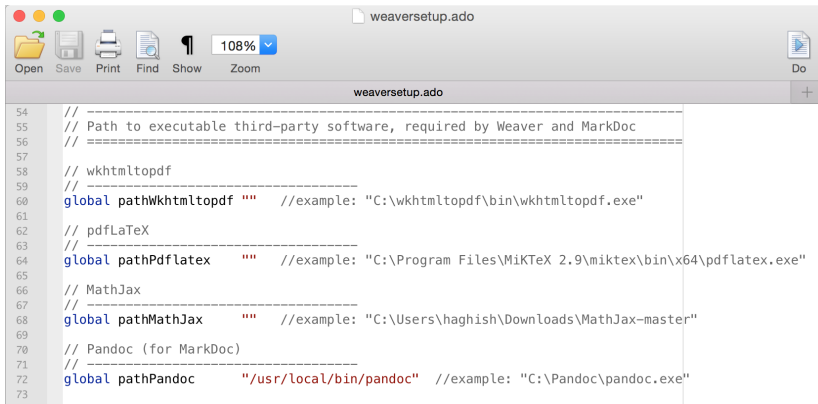
Figure 16: `github search` output

# Third-party software installation (optional)

- ▶ Previous versions of `markdoc` required other software for generating Word and PDF documents. However, in the recent version of `markdoc`, **this is no longer a necessity**. The third-party software are particularly required for `Stata` version 14 and below.
- ▶ Throughout this presentation, I will use the `mini` engine that allows `markdoc` to run independent of any third-party software
- ▶ Nevertheless, installing the third-party software can enhance `markdoc`'s capabilities and is generally recommended
- ▶ The third-party software are
  - ▶ Pandoc for converting Markdown to other file formats
  - ▶ `wkhtmltopdf` for creating PDF documents from source written with Markdown or HTML
  - ▶ users who wish to write with LaTeX will require a LaTeX distribution

- ▶ Pandoc software can be downloaded from `www.pandoc.org` website
  - ▶ Once Pandoc is installed, the path to executable Pandoc on the operating system can be provided to markdoc using the `pandoc(str)` option
- ▶ wkhtmltopdf software can be downloaded from `www.wkhtmltopdf.org`
  - ▶ Next, the path to the executable wkhtmltopdf file should be provided to markdoc using the `printer(str)` option
- ▶ For compiling LaTeX to PDF, a proper LaTeX distribution based on the operating system should be downloaded from `www.latex-project.org`
  - ▶ the path to executable pdfLaTeX compiler should be given to `printer(str)` option.
- ▶ The path to Pandoc, wkhtmltopdf, and pdfLaTeX can be permanently defined using the `weave setup` command.

. weave setup



```

54 // -----
55 // Path to executable third-party software, required by Weaver and MarkDoc
56 // =====
57
58 // wkhtmltopdf
59 // -----
60 global pathWkhtmltopdf "" //example: "C:\wkhtmltopdf\bin\wkhtmltopdf.exe"
61
62 // pdfLaTeX
63 // -----
64 global pathPdflatex "" //example: "C:\Program Files\MiKTeX 2.9\miktex\bin\x64\pdflatex.exe"
65
66 // MathJax
67 // -----
68 global pathMathJax "" //example: "C:\Users\haghish\Downloads\MathJax-master"
69
70 // Pandoc (for MarkDoc)
71 // -----
72 global pathPandoc "/usr/local/bin/pandoc" //example: "C:\Pandoc\pandoc.exe"
73

```

Figure 17: defining the paths to required software permanently

## Automatic installation of third-party software

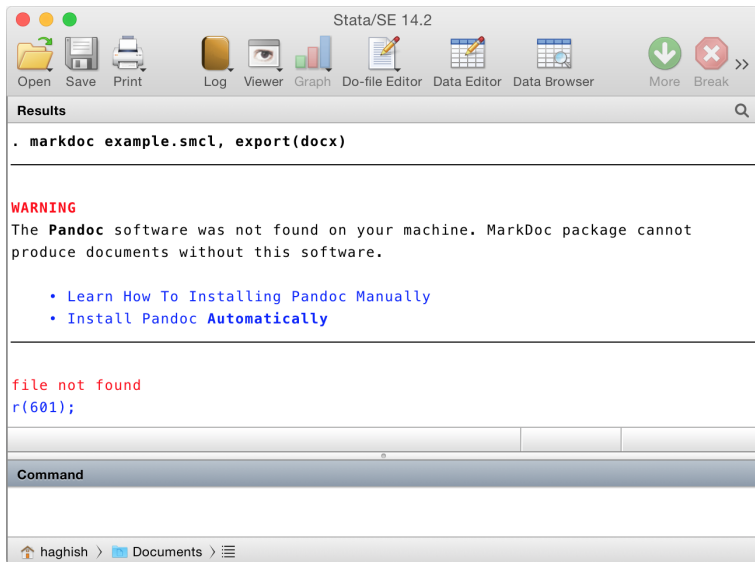
The `markdoc` command includes the `install` option which downloads Pandoc and wkhtmltopdf software automatically, if they are not already installed or cannot be accessed by `markdoc`. As shown in the example below, adding the `install` option will avoid any error regarding the required software and installs them on the fly:

```
qui log using example.smcl, replace  
display "If necessary, install the required software on the fly"  
qui log c
```

```
markdoc example.smcl, export(pdf) install
```



UiO: If the `install` option is not specified and `markdoc` does not detect the required software on your machine, a message will be returned on your machine to indicate that the required software was not found.



Uio: Clicking on the install pandoc automatically will install Pandoc on your machine:

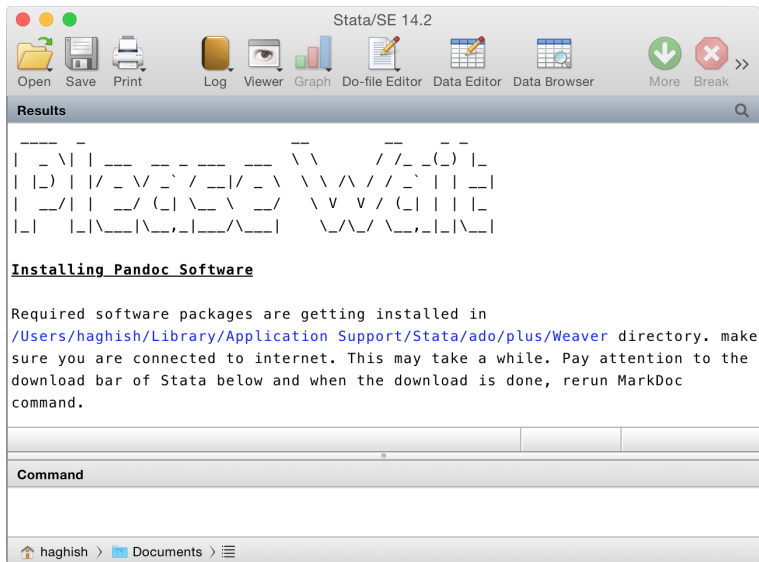
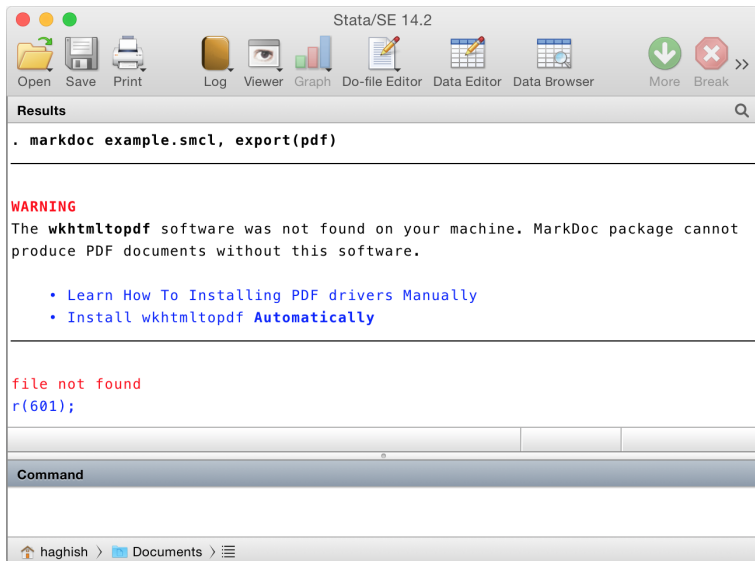


Figure 19: installing Pandoc automatically

UiO: A similar message is displayed if you are exporting a PDF document and `markdoc` does not access `wkhtmltopdf`

`markdoc example.smcl, export(pdf)`



# Workflow

- ▶ `markdoc` has 2 separate modes
- ▶ Passive mode (allows real-time documentation)
  - ▶ Takes a log-file / script file (.ado, .mata, etc.)
  - ▶ It does **NOT** evaluate the code nor reproduce the analysis
  - ▶ It produces a document very fast
- ▶ Active mode (for testing the whole code in a fresh environment)
  - ▶ Takes a do-file
  - ▶ Executes the analysis
  - ▶ Evaluates its reproducibility
  - ▶ It is much slower than the passive mode, because it repeats the analysis

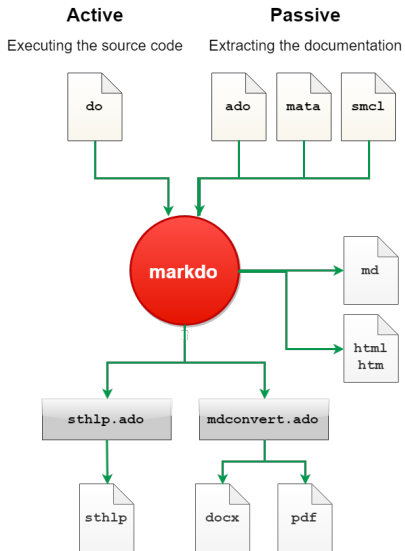


Figure 21: markdoc workflow

# Active documentation

the do-file must be examined in a clean workspace, where no data is loaded in Stata. `markdoc` takes care of such a test, when executed actively.

- ▶ using a single command to convert a `smcl` log-file to various document formats is convenient, but it does not ensure the reproducibility of the source code
- ▶ For example, users might have made changes to the data that are not included in the do-file, but are registered in the log.
- ▶ There are markers for temporarily deactivating the log file. . .
- ▶ Active documentation is more strict, although time-consuming because every time `markdoc` is executed, the whole project is computed again.

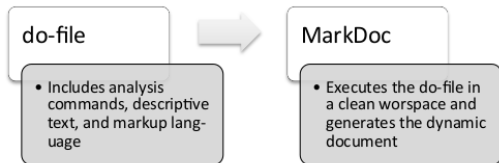


Figure 22: The process of producing dynamic documents with `markdoc`

Let's assume that we have a do-file that only in displays the hello world text in a do-file named *example1.do*:

```
. display "Hello World"
```

Then the dynamic document can be produced by actively executing the do-file as shown below:

```
. markdoc example.do, mini export(docx)
```



Let's have a closer look. We will load a data set in Stata. Then we execute the command related to the loaded data set with `markdoc`. We would expect an **error**, because in the workspace that `markdoc` is using to test the reproducibility of the code, there is no information about the loaded data set.

- ▶ we load the *Auto* data set  
    `. sysuse auto, clear`
- ▶ we create a do file that simply displays the first line of the data. we name the file *example2.do* and execute it in Stata:

```
. do example2.do
```

```
. list in 1
```

```

+-----+
1. | make          | price | mpg | rep78 | headroom | trunk |
   | AMC Concord | 4,099 | 22 |      3 |      2.5 |    11 |
   +-----+
   | weight  | length | turn | displa~t | gear_r~o |
   | 2,930  |    186 |   40 |      121 |     3.58 |
   +-----+
   |                                     |
   |                               foreign |
   |                               Domestic |
   +-----+

```

But if we examine it with `markdoc`, we get the following error.  
`markdoc` says it can't find the data!

```
. markdoc example2.do, mini export(pdf)
```

```
. list in 1
```

```
observation numbers out of range
```

```
r(198);
```

```
end of do-file
```

```
r(198);
```

# Passive documentation

- ▶ Is used for generating help files, package vignettes, or quick analysis documents from a log-file
- ▶ the SMCL log-file registers every entry in Stata including comments, commands, and text-based output, `markdoc` can produce a dynamic document passively from the SMCL log-file.
- ▶ This workflow is indeed convenient, but not recommended for generating analysis documents
- ▶ the log-file – which is updated in real-time during the analysis session – can be used to generate the document in real-time too

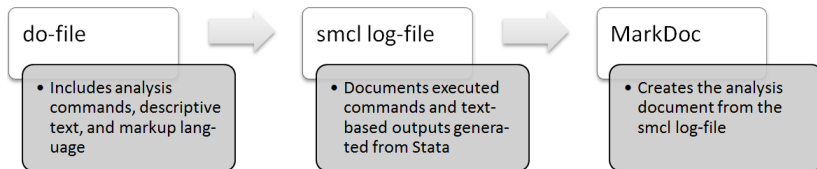


Figure 23: The process of producing dynamic documents with `markdoc`

# Example

Create a do file with this code and generate a PDF document with syntax highlighter. name the example *example3.do*. let's also use a few of the `markdoc` options to create the title of the document.

```
. quietly log using example, replace smcl  
  
. display "Hello World"  
Hello World  
  
. qui log c  
. markdoc example.smcl, mini export(pdf) statax
```

# Syntax

To produce a dynamic document, the *filename* of the documentation source should be given to `markdoc`

- ▶ **PASSIVE MODE:** a smcl log file with `.smcl` file extension or a script file with `.ado` or `.mata` extension
- ▶ **ACTIVE MODE:** a do-file with `.do` extension
- ▶ If the file extension is not specified, **SMCL log file is assumed** - Specifying the file extension is recommended to provide further clarity

# Essential syntax

```
markdoc filename [, options]
```

Option	Description
<code>mini</code>	runs <code>markdoc</code> independent of any third-party software
<code>install</code>	installs Pandoc and Wkhtmltopdf software, if not found
<code>export(name)</code>	format; it can be <code>docx</code> , <code>pdf</code> , <code>html</code> , <code>sthlp</code> , <code>slide</code> , <code>md</code> , or <code>tex</code>
<code>replace</code>	replaces exported document, if exists
<code>statax</code>	activates <i>Statax</i> (Haghish 2019b) syntax highlighter
<code>helplayout</code>	appends a Markdown help layout template to a script file

Figure 24: markdoc's essential syntax



# Markup Languages

- ▶ markdoc supports
  - ▶ LaTeX (requires third-party software)
  - ▶ HTML
  - ▶ Markdown
- ▶ In this lecture we will focus on Markdown, which is the simplest. The following links, from its developer's site, can provide a good background about Markdown: -  
<https://daringfireball.net/projects/markdown/> -  
<https://daringfireball.net/projects/markdown/syntax> -  
<https://daringfireball.net/projects/markdown/dingus>
- ▶ Markdown is
  - ▶ minimalistic and clean
  - ▶ simple to read and write
  - ▶ helps to focus on the content
  - ▶ can be converted to many formats
  - ▶ it has become the standard documentation markup language

- ▶ in `markdoc` the documentation (Markdown, html, or LaTeX) are written within a special comment signs.

```
/**
```

```
...
```

- ▶ There is no limit in how many times you can place these signs in a do-file.
- ▶ These signs can appear anywhere in the analysis document, but not inside loops - I will introduce the `txt` command later on which can be included inside loops and programs

Markdown syntax	Result
Heading 1 =====	Heading 1
Heading 2 -----	Heading 2
###Heading 3	Heading 3
####Heading 4	Heading 4
plain text paragraph	plain text paragraph
> text	block quote
<b>**Bold**</b> or <b>__Bold__</b> text	<b>Bold</b> or <b>Bold</b> text
<i>*Italic*</i> or <i>_Italic_</i> text	<i>Italic</i> or <i>Italic</i> text
`monospace` text	monospace text
superscript <sup>2</sup>	superscript <sup>2</sup>
---	horizontal rule
1. Ordered item1 A. Sublist 1 a. Subsublist 1 2. Ordered item2	1. Ordered item1 A. Sublist 1 a. Subsublist 1 2. Ordered item2
* Unordered item1 * Sublist 1 * Subsublist 1 * Unordered item2	• Unordered item1 - Sublist 1 * Subsublist 1 • Unordered item2
![Text](filename)	Insert an image with description
[Text](http://url)	Insert a hyperlink

Figure 25: Markdown syntax

# Additional Markup Notations

- ▶ Additional markers further organize documents prepared for `markdoc` software
- ▶ additional markers fall into two categories
  - ▶ Passive markers, used for writing static text and styling
  - ▶ Active markers, used for interpreting Stata macros in the document
- ▶ The Active markers only work if `markdoc` is executed in the active mode

# Passive markers

- ▶ annotate Stata “commands” and “outputs”
- ▶ they help to write a clean analysis report
- ▶ By default, MarkDoc includes all of the do-files and text outputs that appear in the Stata results windows. The additional notations allows you to be more selective about what to include:
  - ▶ Hiding Stata commands
  - ▶ Hiding Stata output
  - ▶ Hiding a part of a do-file
  - ▶ Importing external files

Table 2: Additional Notation Markers

Marker	Description
<code>/**/</code>	Exclude the Stata command but keep the output
<code>/***/</code>	Exclude the Stata output but include the command
<code>//OFF</code>	Exclude everything in the log that follows
<code>//ON</code>	Deactivate the <code>//OFF</code> marker
<code>//IMPORT <i>filename</i></code>	Include an external file (Markdown, HTML, LaTeX)

Figure 26:

# Hiding Stata commands

```
// ----- Beginning additional_hide.do -----
```

```
/**
```

```
Hiding Stata commands
```

```
-----
```

The command bellow will not appear in the dynamic document.  
However, their output will be included.

```
/**/ sysuse auto, clear
```

```
/**/ summarize
```

Uio: Executing the `markdoc` command will results in the following output:

```
. markdoc additional_hide.do, export(docx)
```

## Hiding Stata commands

The command bellow will not appear in the dynamic document. However, their output will be included.

```
(1978 Automobile Data)
```

Variable	Obs	Mean	Std. Dev.	Min	Max
make	0				
price	74	6165.257	2949.496	3291	15906
mpg	74	21.2973	5.785503	12	41
rep78	69	3.405797	.9899323	1	5
headroom	74	2.993243	.8459948	1.5	5
trunk	74	13.75676	4.277404	5	23
weight	74	3019.459	777.1936	1760	4840
length	74	187.9324	22.26634	142	233
turn	74	39.64865	4.399354	31	51
displacement	74	197.2973	91.83722	79	425
gear_ratio	74	3.014865	.4562871	2.19	3.89
foreign	74	.2972973	.4601885	0	1

Figure 27:



# Hiding Stata output

```
// ----- Beginning additional_hide2.do -----
```

```
/**
```

```
Hiding Stata output
```

```
-----
```

```
/**/ sysuse auto, clear
```

```
/**/ summarize
```

```
. markdoc additional_hide2.do, export(docx)
```

## Hiding Stata output

```
. sysuse auto, clear  
. summarize  
|
```

Figure 28:

# Hiding a part of a do-file

- ▶ MarkDoc also allows hiding a section of the do-file, without influencing the code execution

```
// ----- Beginning additional_hide3.do -----
```

```
/**
```

```
Hiding Stata commands and output
```

```
-----
```

```
//OFF
```

```
sysuse auto, clear  
summarize
```

```
//ON
```

# Importing external files

- ▶ A convenient feature for producing sophisticated documents
  - Slides - Handouts - eBook!
- ▶ It reads other files (tables, documents, etc) into the main document
- ▶ This is the feature you are most-likely looking for writing publication-ready documents

## Example

- ▶ create a text file and name it *Intro.txt*
- ▶ Import the text file passively into a do-file
- ▶ execute `markdoc` and create a PDF file

```
Intro.txt
```

```
-----
```

As shown in this example, the text that is written in  
\_\_`intro.txt`\_\_ will appear in the final document.

The

```
// ----- additional_import2.do -----
```

```
//IMPORT intro.txt
```

```
. markdoc additional_import2.do, export(pdf)
```

## **Intro.txt**

As shown in this example, the text that is written in **intro.txt** will appear in the final document.

Figure 29: Preview of the output document

# estout package for exporting LaTeX tables

- ▶ LaTeX also has a command for including external tex files.
- ▶ we will use the `estout` package for generating a publication-ready better table
  - . `ssc install estout`
- ▶ In the next example, first a LaTeX table is exported from Stata
- ▶ Then we write a simple LaTeX document and allow `markdoc` to complete the LaTeX layout automatically



```
// ----- Beginning additional_import.do -----
```

```
//OFF
```

```
sysuse auto, clear
```

```
sysuse auto
```

```
eststo: quietly regress price weight mpg
```

```
eststo: quietly regress price weight mpg foreign
```

```
esttab using table.tex, replace
```

```
eststo clear
```

```
//ON
```

```
/**
```

```
\section{Including external file}
```

```
\input{table.tex}
```

```
. markdoc additional_import.do, markup(latex) export(pdf) n
```

## 1 Including external file

	(1)	(2)
	price	price
weight	1.747** (2.72)	3.465*** (5.49)
mpg	-49.51 (-0.57)	21.85 (0.29)
foreign		3673.1*** (5.37)
_cons	1946.1 (0.54)	-5853.7 (-1.73)
<i>N</i>	74	74

*t* statistics in parentheses

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

Figure 30: Preview of the PDF document

# Active markers

- ▶ used for writing dynamic text, which includes *scalars* or *macros* that should be automatically interpreted into the text
- ▶ only work in the Active mode
- ▶ can show the values of
  - ▶ scalars
  - ▶ variable observations
  - ▶ local macro
  - ▶ global macros
- ▶ values should be placed within `<!*!>` marker

## ACTIVE MARKERS TABLE

Object	Description
<code>&lt;!scalar!&gt;</code>	Numeric or String scalar
<code>&lt;!matrix[r,c]!&gt;</code>	Numeric scalar from a matrix
<code>&lt;!variable[n]!&gt;</code>	Nth observation of a variable
<code>&lt;!`local'!&gt;</code>	Numeric local macro
<code>&lt;!\$global!&gt;</code>	Numeric global macro
<code>&lt;!"`local'"!&gt;</code>	String local macro
<code>&lt;!"\$global"!&gt;</code>	String global macro

Figure 31: Preview of the PDF document

# Example

```
local a = 1
scalar b = 2
matrix define A = (20,30\40,50)
```

```
/**/ di as txt "> " _n ///
`"> This is heading "' `a' _n ///
`"> ===== "' _n ///
`"> "' _n ///
`"> The values of a matrix can be displayed within the text. For example,"' _n
`"> you can write "' A[1,1] " which shows the scalar of the first row and" _n
`"> first column of the matrix in your documentation. This feature makes"' _n /
`"> writing dynamic text much more convenient compared to the previous procedur
`"> "' _n ///
`"> This is heading "' b _n ///
`"> ----- "' _n ///
`"> "' _n ///
`"> REMEMBER, that this procedure only works if you execute a do-file with"' _n
`"> markdoc, that is, using the `markdoc filename.do, export(format)` syntax."
```

## Additional commands

- ▶ these commands are borrowed from `weaver` package - they are installed automatically as a dependency
- ▶ They come very handy when the document is generated by a program dynamically or within a loop
- ▶ They allow more details for styling a document, compared to Markdown - Adding a figure Automatically - adding a dynamic table - adding dynamic text

# Adding figures dynamically

- ▶ we previously used Markdown to include an image in the document
- ▶ The process was:
  1. saving a graph from Stata to the disk
  2. including the graph to the dynamic document
- ▶ This procedure can be further simplified, using the `img` command
  1. Automatically capture the current graph from Stata and include it in the dynamic document
  2. Include a figure from the disk/internet in the dynamic document
  3. Resize the width and the height of the image in the dynamic document
  4. Align the image to the left (default) or center of the document
  5. Add a graph description

# Syntax of `img` command

Import graphical files in the dynamic document

```
img [using filename] [, markup(str) title(str) width(int) height(int) left c
```

Automatically include the current graph from Stata in the dynamic document

```
img [, markup(str) title(str) width(int) height(int) left c
```



# Examples

- ▶ create a do-file and execute it with `markdoc` actively or passively

```
. sysuse auto  
. histogram price  
. img
```

In this example, `img` has stored the current graph in a directory called **Weaver-figure**

# Adding text dynamically

- ▶ the `txt` command is somehow like the `display` command, but it's used for writing text in the dynamic document
- ▶ it can be used to write text within loops or programs and interpret scalars, global, or local macros within
- ▶ try typing `txt 1+1`

```
. sysuse auto  
. summarize price  
. txt "the mean of Price variable is " r(mean)
```

# Syntax of the `txt` command

```
txt [code] [display_directive [display_directive [...]]]
```

where the `display_directive` can be:

```
"double-quoted string"  
~"compound double-quoted string"  
[%fmt] [=]exp  
_skip(#)  
_column(#)  
_newline[(#)]  
_dup(#)  
,  
,,
```

# Writing dynamic tables

- ▶ `tbl` simplifies writing and styling dynamic tables
- ▶ The default markup language is Markdown, but it also support LaTeX and HTML
- ▶ It can align the content of each column to the left, center, or right
- ▶ It creates a table somehow similar to the way a matrix is defined in Stata

# tbl Syntax

The syntax of the command is:

```
tbl (*[,*...] [\ *[,*...] [\ [...]]]) ///  
    [, markup(str) title(str) width(int) height(int) c
```

where the \* represents a display directive, which is:

```
"double-quoted string"  
`"compound double-quoted string"`  
[%fmt] [=]exp  
,  
{l}  
{c}  
{r}
```

# Examples

- ▶ creating a simple 2x3 table with string and numbers

```
tbl ("Column 1", "Column 2", "Column 3" \ 10, 100, 1000 )
```

- ▶ creating a table that includes scalars and aligns the columns to left, center, and right respectively

```
tbl ({l}"Left", {c}"Centered", {r}"Right" \ c(os), c(machine_type), c(username)
```

# Dynamic Presentation Slides

- ▶ `markdoc` supports generating presentation slides in HTML and PDF formats
- ▶ slides can be made from the same source used for generating analysis documents
- ▶ the main difference is that slides should be broken into small frames
- ▶ There are two possibilities for separating the frames:
  - ▶ using header 1 e.g. `# Header 1`
  - ▶ using horizontal line syntax `- - -`

type `db markdoc` and check out the **Presentation Slide** tab

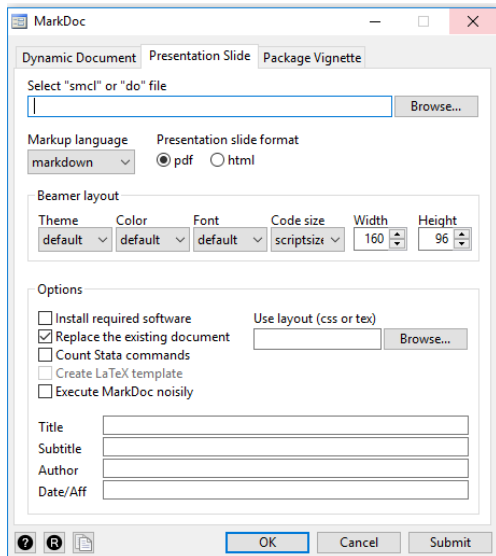


Figure 32: Using `markdoc` GUI for generating slides



```
quietly log using example, replace smcl
```

```
/***
```

```
Using `markdoc` for generating slides
```

```
=====
```

```
> Let's begin by using some Stata commands
```

```
sysuse auto, clear
```

```
summarize price
```

```
histogram price
```

```
/***
```

```
---
```

```
img
```

# Stata help files

- ▶ Stata has it's own markup language
  - ▶ *Stata Markup and Control Language* (SMCL)
- ▶ All help files as well as default log files are written in this markup language
- ▶ Writing documentation with SMCL is not appealing:
  1. smcl is difficult
  2. somehow messy to write
  3. difficult to read, write, and comprehend
- ▶ literate programming with smcl is difficult and makes the script file too complex to read

- ▶ `markdoc` can generate Stata help files from Ado and Mata files
- ▶ The software documentation can be written in Markdown, using the same procedure
- ▶ If the documentation can be exported to Stata help files or package vignette
- ▶ Type `db markdoc` and navigate to the **Package Vignette** tab:

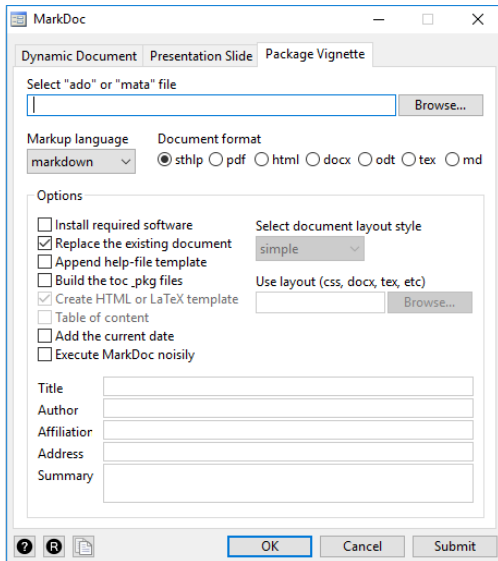


Figure 33: Using markdoc GUI for generating software documentation

# Example

- ▶ Let's make an Ado file, and use some simple Markdown syntax to write in it
- ▶ let's write a:
  - ▶ Header 1
  - ▶ Header 2
  - ▶ style some text
  - ▶ Indent text
  - ▶ add a line
  - ▶ add a link

```
/**
Title
=====
```

`__commandname__` - explain your command briefly. You can use simplified syntax to make text `_italic_`, `__bold__`, `***emphasized***`, or add `[hyperlink]` (<http://www.haghighi.com/markdoc>)

```
Syntax
-----
```

```
> __XXX__ _varlist_ [, _options_]
```

```
Example(s)
-----
```

```
    explain what it does
        . example command
```

```
    second explanation
        . example command
```

- ▶ execute this example with `markdoc` GUI and generate:
  - ▶ a `sthlp` file
  - ▶ a `html` vignette
  - ▶ a `docx` vignette
- ▶ In the GUI, there is an option for appending documentation to an `Ado` file
- ▶ Apply the **Append help-file template** to see an example documentation template
- ▶ generate a `sthlp` and `html` file from the template

```

Title

commandname — explain your command briefly.  You can use simplified syntax to make
text italic, bold, emphasized, or add hyperlink

Syntax

    XXXX varlist =exp [if] [in] [weight] using filename [, options]

options

-----

minabbrev: description of what option
breakline: break each line with adding 2 space bars
minabbrev(arg): description of another option

-----

by is allowed: see \[D\] by
fweight is allowed: weight

Description

XXXX does ... (now put in a one-short-paragraph description of the purpose of the
command)

Options

whatever does yak yak

    Use > for additional paragraphs within and option description to indent the
    paragraph.

    2nd option etc.

Remarks

The remarks are the detailed description of the command and its nuances. Official
documented Stata commands don't have much for remarks, because the remarks go in the
documentation.

Example(s)

explain what it does
    . example command

second explanation
    . example command
  
```

Figure 34: Example help file template



# References

- ▶ Garfield, J. (1995). How students learn statistics. *International Statistical Review / Revue Internationale de Statistique*, 63 (1), 25-34. Retrieved from <http://www.jstor.org/stable/1403775>
- ▶ Baloglu, M. (2003). Individual differences in statistics anxiety among college students. *Personality and Individual Differences*, 34 (5), 855-865.
- ▶ Onwuegbuzie, A. J. (2004). Academic procrastination and statistics anxiety. *Assessment & Evaluation in Higher Education*, 29 (1), 3-19.
- ▶ Loscalzo, J. (2012). Irreproducible experimental results: Causes, (mis)interpretations, and consequences. *Circulation*, 125 (10), 1211-1214. Retrieved from <http://circ.ahajournals.org/content/125/10/1211.short> doi: 10.1161/CIRCULATIONAHA.112.098244
- ▶ Baggerly, K. A., & Berry, D. A. (2009). Reproducible research.
- ▶ Peng, R. D. (2011). Reproducible research in computational science. *Science* (New York, Ny), 334 (6060), 1226.
- ▶ Stodden, V., Leisch, F., & Peng, R. D. (2014). Implementing reproducible research. CRC Press.
- ▶ Gentleman, R., & Lang, D. T. (2012). Statistical analyses and reproducible research. *Journal of Computational and Graphical Statistics*.

- ▶ Knuth, D. E. 1983. The WEB system of structured documentation. Technical Report STAN-CS-83-980, Department of Computer Science, Stanford University. <http://infolab.stanford.edu/pub/cstr/reports/cs/tr/83/980/CS-TR-83-980.pdf>
- ▶ Xie, Yihui, Joseph J. Allaire, and Garrett Grolemond (2018). R markdown: The definitive guide. CRC Press.
- ▶ Haghish E. F. (2016). Markdoc: Literate Programming in Stata. The Stata Journal, 16(4):964-988. doi:10.1177/1536867X1601600409
- ▶ Haghish, E. F. (2020). Software documentation with markdoc 5.0. The Stata Journal, 20(2), 336-362.
- ▶ Open Science Collaboration. (2015). Estimating the reproducibility of psychological science. Science, 349(6251).
- ▶ Iso-Ahola, S. E. (2017). Reproducibility in psychological science: When do psychological phenomena exist?. Frontiers in Psychology, 8, 879.

- ▶ Ioannidis, J. P. (2005). Why most published research findings are false. PLoS medicine, 2(8), e124.
- ▶ Zunger, J. (2018). Computer science faces an ethics crisis. The Cambridge Analytica scandal proves it. Boston Globe, 22.