Systematic management of research materials with Git and GitHub

E. F. Haghish

University of Göttingen

Overview

- Part 1: Introduction to version control software (Git & GitHub)
  - Version control software
    - Different version control models
  - Git software
  - GitHub website
- Part 2: Using GitHub for hosting code, data, manuscripts, documentation, and web content
  - Publishing Stata/R software on GitHub
  - The `github` Stata package
    - Searching, installing, and managing Stata packages
    - Building package installation files
  - Publishing on GitHub
    - Software documentation
    - Data analysis code
    - Data
    - Manuscripts, etc.
  - Collaborating via GitHub

Criteria for a discussion

- What are our demands from a perfect open-science platform?
  - What features do we need?
  - How these features change across disciplines?
  - How easy would it be to integrate such a platform in classroom for education?
- Points to consider
  - Functionality of Git and GitHub
  - Familiarity / Learning curve
  - Scalability
  - Openness
  - Sustainability
  - Community
  - Support
  - Costs
  - Efficiency

Version control software

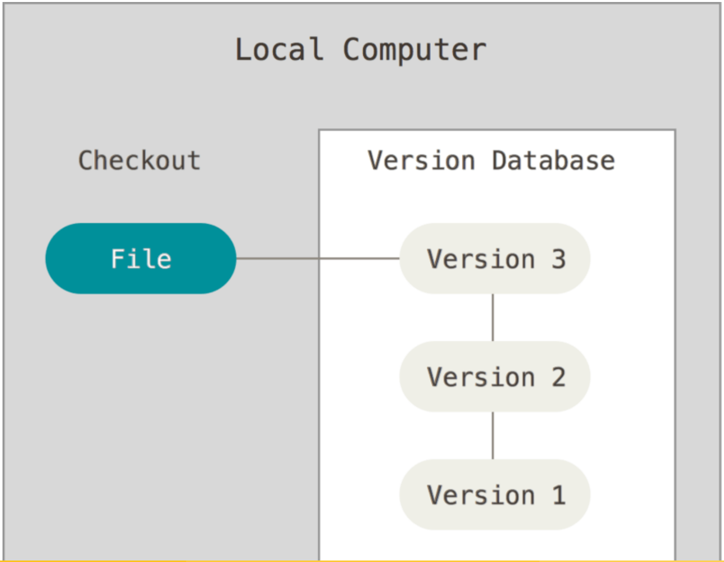
- Version control software are used everyday (backing up smartphones, computers, etc)
- A version control software documents changes made to files
    - Helps with recalling specific versions later
- It is not limited to programming code; changes made to most types of files can be monitored
- It is extensively used for individual work as much as teamwork
- Examples of different usages
    - Web designers
    - Writers
    - A programmer
        - What updates caused a problem
    - A team of programmers
        - What update caused a problem, who introduced the error, when, and in which part of the code
    - Backing up the project at each step
        - If you ruin a file or remove it accidentally you can back it up

Version control software

- Several version control software exist, some made by Microsoft, IBM, Autodesk, etc.
- Version control software have different architecture models
    - Local version control
    - Client-server model (Centralized Version Control System), where only a single repository exists on the server for all users
        - Users do not have a local clone of the project
        - users need internet access
        - Hierarchical collaboration within groups is not possible
    - Distributed model, where every user works on his own copy of the repository
        - Can be extended for users without a writing access to the original repository
        - Clients fully mirror the repository, including its full history
        - Every clone is really a full backup of all the data
        - Hierarchical workflow can be planned, collaboration can be within groups

Local version control

- Built-in within many operating systems for backup
- Useful for individual work
- Not useful for collaborative work



Client-server model (Centralized)

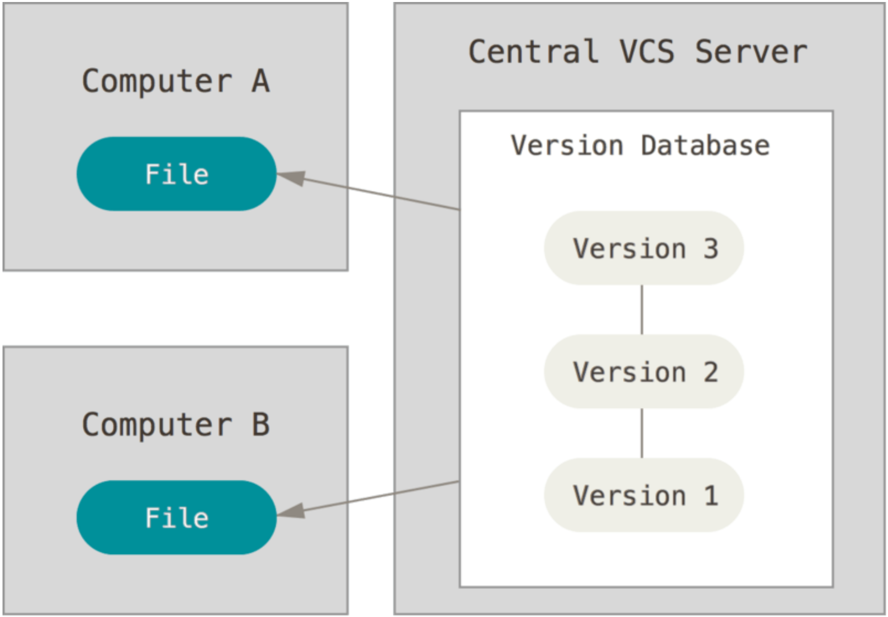


Figure 2: Client-server version control (from **git-scm.com**)

Distributed Version Control Systems

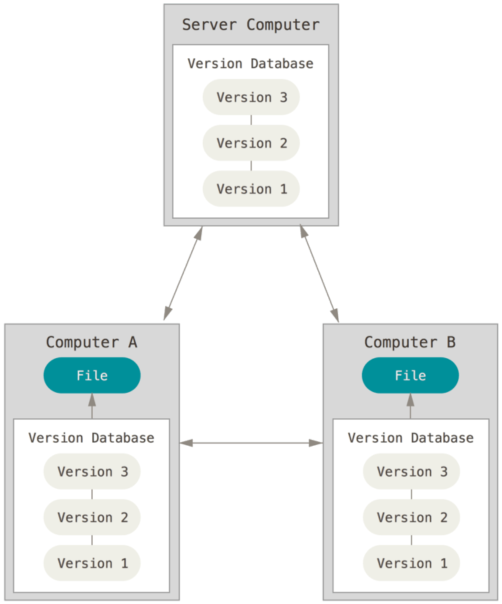


Figure 3: Distributed version control (from **git-scm.com**)

Git

- Git is developed by *Linus Torvalds* in 2005 to meet the demands of maintaining Linux Kernel
  - They used *BitKeeper* vesion control software prior to developing Git
  - They needed a fast version control that would be
    - fully based on distributed model
    - able to handle large projects
    - able to handle non-linear development, where a high number of branches evolve in parallel
- Git is a Distributed Version Control (DVC) system
- The *git-scm.com* website provide plenty of free resources about Git
- Working with Git has a learning curve
  - Git is based on Command line interface (CLI)

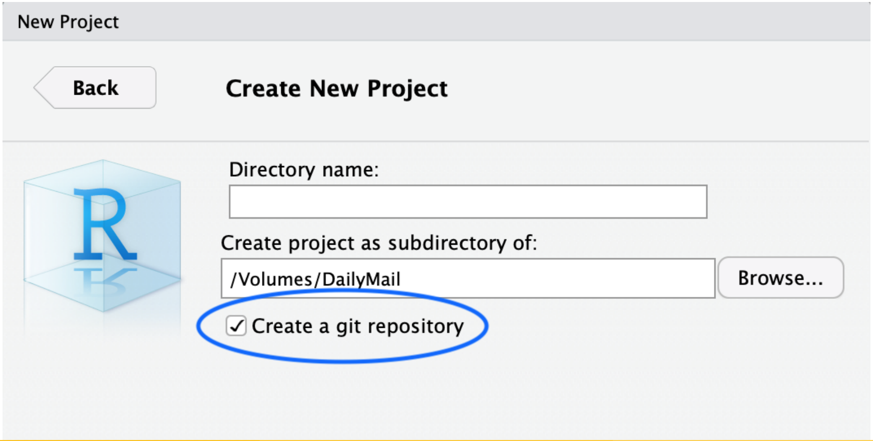
Git

- You clone the repository locally, which is a full backup of the project
- To update the repository, you do not need internet. Everything is stored locally.
- Once the code is updated, you *commit* it, to register the changes in the local database
- Collaborating via Git requires a server; Git is not a server itself
    - To merge the new changes you have made, make a *pull request*
    - This is the only part where internet connection is needed
- Download Git from https://git-scm.com/downloads

Git GUI

Using a graphical user interface can greatly help with working with Git. The GitHub application can be used for managing files locally

- Rstudio (Mac, Windows, Linux)
    - Ideal for managing data analysis and documentation within a version control
- SmartGit (Mac, Windows, Linux)
- GitHub application for Mac and Windows

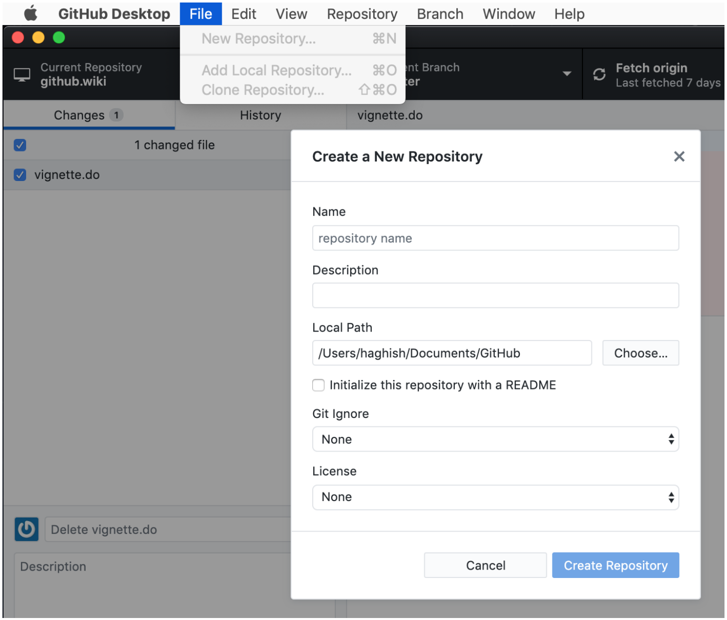


Directory name:

Create project as subdirectory of: /Volumes/DailyMail

✓ Create a git repository

New repository



Create a New Repository

Name: repository name

Description:

Local Path: /Users/haghish/Documents/GitHub [Choose...]

☐ Initialize this repository with a README

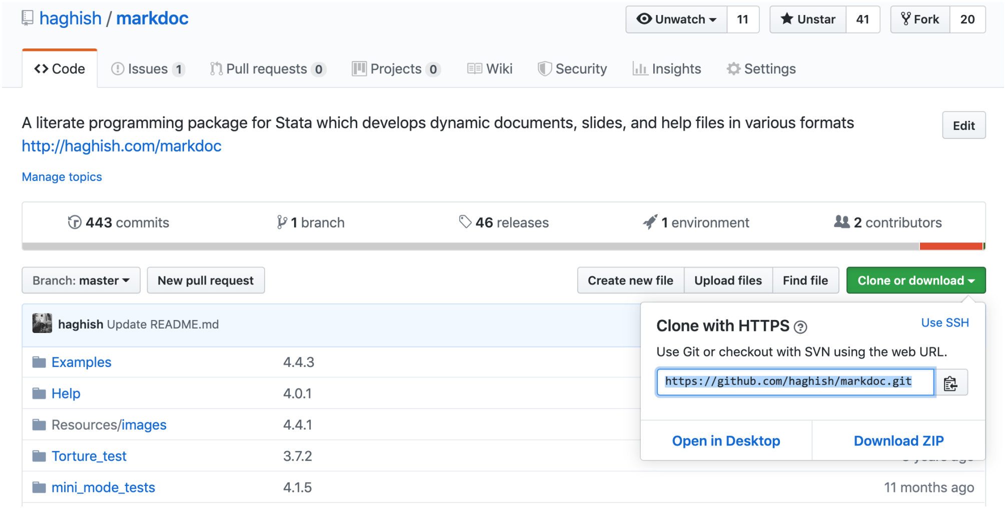
Git Ignore: None

License: None

Cancel | Create Repository

Figure 5: Creating a new repository for a project

Cloning existing repository


haghish / markdoc

A literate programming package for Stata which develops dynamic documents, slides, and help files in various formats

http://haghish.com/markdoc

443 commits | 1 branch | 46 releases | 1 environment | 2 contributors

haghish Update README.md

Examples	4.4.3	
Help	4.0.1	
Resources/images	4.4.1	
Torture_test	3.7.2	
mini_mode_tests	4.1.5	11 months ago

Clone with HTTPS

Use Git or checkout with SVN using the web URL.

```
https://github.com/haghish/markdoc.git
```

Figure 6: Clone a repository with a URL

Committing/discarding changed files

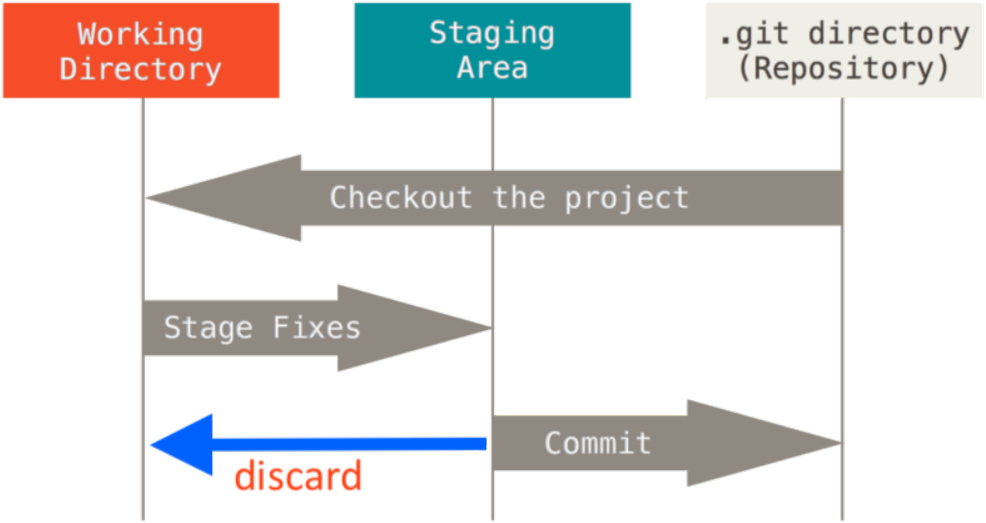


Figure 7: Staging to committing or discard changes

Committing/discarding changed files

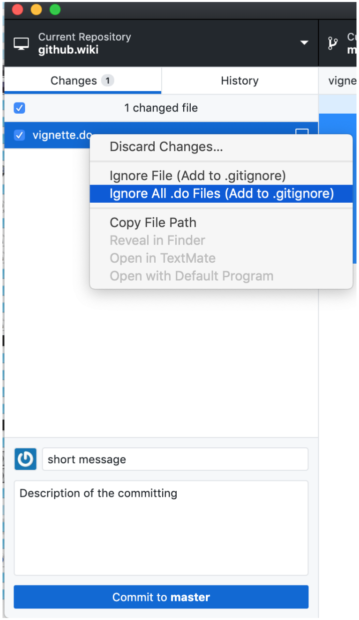


Figure 8: Staging to committing or discard changes

Learn more about Git

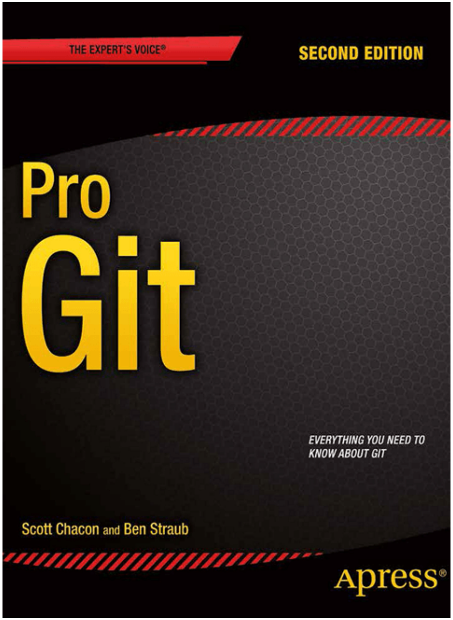

THE EXPERT'S VOICE®
SECOND EDITION
Pro Git
EVERYTHING YOU NEED TO KNOW ABOUT GIT
Scott Chacon and Ben Straub
Apress®

Figure 9: Pro Git eBook is available for free from **git-scm.com**

GitHub

- GitHub is a social coding site that offers plenty of features for collaboration on software such as
  - tracking issues
  - documentation platform
  - managing tasks
  - Git version control
- the largest host for Git repositories and also the largest code hosting site
- The preeminent advantage of GitHub is its social nature.
  - GitHub is a combination of Git with a social media
  - developers broadcast their coding exercise
  - follow others' activities
  - audit a repository
  - discover recent projects
  - collaborate
  - the pro-social characteristics of GitHub promotes project dissemination
  - peer-reviewing the code

GitHub

- GitHub repositories can be private or public
- GitHub utilizes a *pull-based development model*
    - it permits anyone to view, fork, and contribute to any public repository on GitHub
    - The pull-based development model relies on a DVC for tracking changes and contributions
- Contributing to a project via GitHub happens in two ways
    - Direct change, for those who have writing access to the repository
    - or by *forking* the repository, creating a copy of the repository
        - Changes to the original repository can be made through submitting a *pull request*
        - If accepted by the repository owner/maintainer, the change will be incorporated in the repository

GitHub is dominated by programmers, but other research fields are rapidly catching up

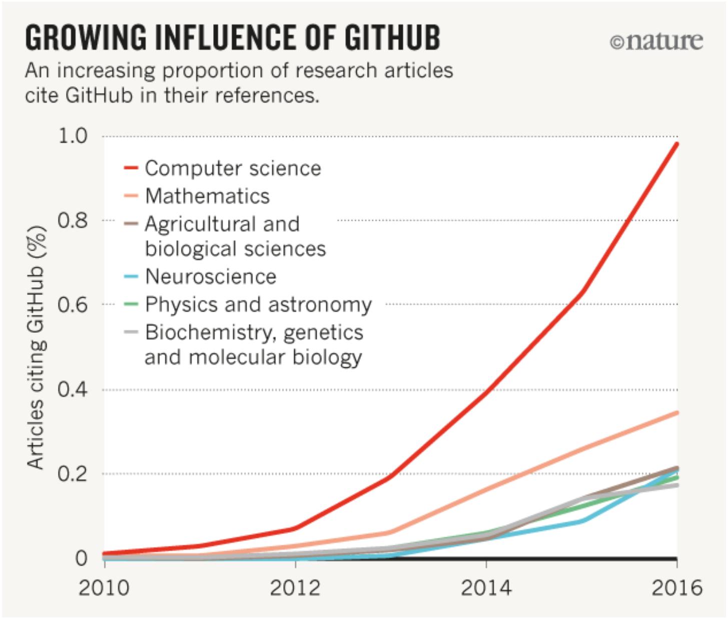


Figure 10: Clone a repository with a URL

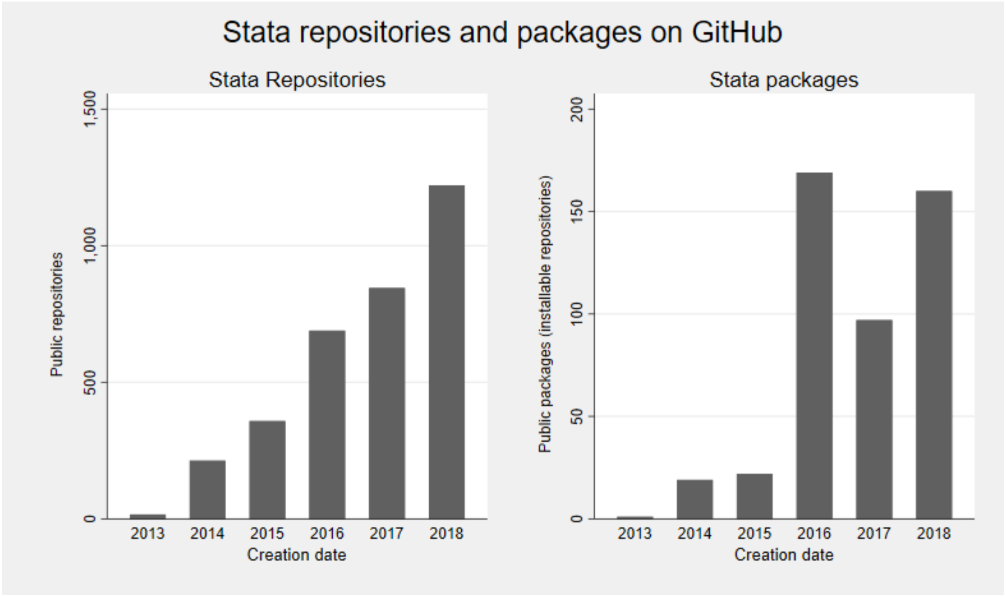


Figure 11: Number of Stata repositories and packages by creation date

Hosting Data on GitHub

- Version control can help with data management
  - e.g. “Open Exoplanet Catalogue”, is a database of all discovered extra-solar planets, hosted on GitHub
- Git is more than enough for text-based data (CSV, XML, JSON, etc.), but has difficulties with binaries
  - Changes (diffs) are not human-readable in binaries
  - Merging edited binary files is even a bigger challenge for Git
- Try to use text-based file formats for the best results
  - e.g. instead of using Stata’s and R’s native data formats, use CSV file formats, if possible
    - Any changed observation or value can can be tracked
- On GitHub, no repository is necessarily permanent and the repo’s owners can take it down.
  - Your clone of the repository will not be removed
  - Use other websites for making permanent URL links with Digital Object Identifiers (DOI) for your publication
    - use https://figshare.org/ or https://zenodo.org/ for permanent URL
    - https://guides.github.com/activities/citable-code/

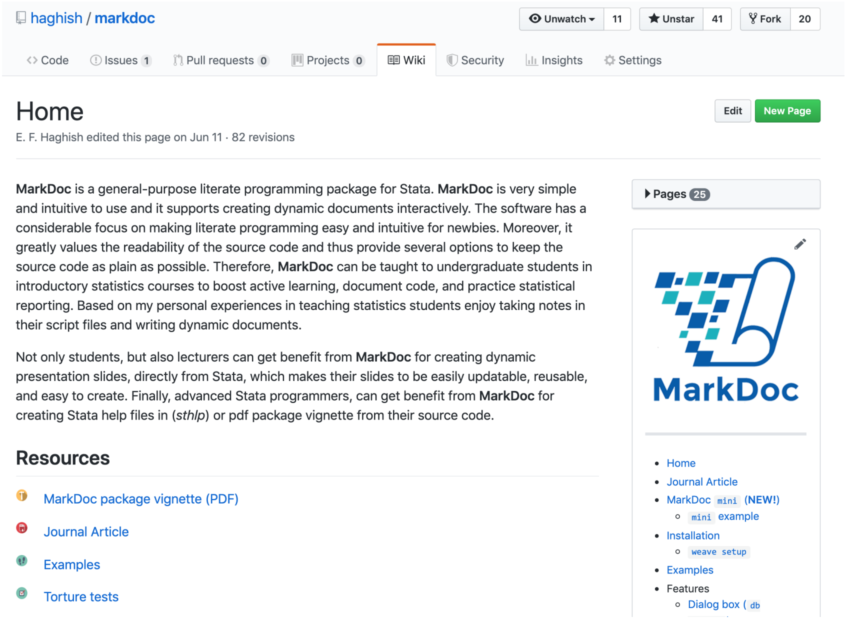
Collaboration on text documents

- GitHub cannot show what has changed in a Microsoft Word Docx file, it only notifies that it has changed
- Many authors publish the LaTeX or Markdown source of their free ebooks via GitHub
- Open formats such as XML or RTF can be viewed on GitHub although their markup annotation is complex and not human readable
- The best results can be obtained with plain text documents such as LaTeX and particularly Markdown
    - Use LaTeX if the document requires a complex layout
    - Use Pandoc to convert Markdown documents to Docx, while applying a complex layout
- Remember that the biggest benefit of GitHub is its social nature; try to keep your files human readable and easy to read for anyone.

Collaborative software documentation on GitHub

- GitHub offers a Markdown-based software documentation platform, called **Wiki**
  - GitHub Wiki can be used for any type of collaborative documentation about the repository
- The documentation is a separate *sub-repository*, that can be cloned by anyone
  - Software documentation can be collaborative
  - Can be updated automatically, by exporting Markdown documentation (e.g. using `markdoc`)
- GitHub also offers a web host for each repository to publish a site

Collaborative software documentation on GitHub


haghish / markdoc

Unwatch 11 · Unstar 41 · Fork 20

Code · Issues 1 · Pull requests 0 · Projects 0 · Wiki · Security · Insights · Settings

Home

Edit · New Page

E. F. Haghish edited this page on Jun 11 · 82 revisions

MarkDoc is a general-purpose literate programming package for Stata. **MarkDoc** is very simple and intuitive to use and it supports creating dynamic documents interactively. The software has a considerable focus on making literate programming easy and intuitive for newbies. Moreover, it greatly values the readability of the source code and thus provide several options to keep the source code as plain as possible. Therefore, **MarkDoc** can be taught to undergraduate students in introductory statistics courses to boost active learning, document code, and practice statistical reporting. Based on my personal experiences in teaching statistics students enjoy taking notes in their script files and writing dynamic documents.

Not only students, but also lecturers can get benefit from **MarkDoc** for creating dynamic presentation slides, directly from Stata, which makes their slides to be easily updatable, reusable, and easy to create. Finally, advanced Stata programmers, can get benefit from **MarkDoc** for creating Stata help files in (*sthlp*) or pdf package vignette from their source code.

Resources

- MarkDoc package vignette (PDF)
- Journal Article
- Examples
- Torture tests

Pages 25

MarkDoc

- Home
- Journal Article
- MarkDoc `mini` **(NEW!)**
  - `mini` example
- Installation
  - `weave setup`
- Examples
- Features
  - Dialog box (`db`

Figure 12: An example of Wiki software documentation

Hosting Statistical code on GitHub

- Many R and Stata users develop their statistical packages on GitHub
- All R packages hosted on CRAN, also exist on GitHub
    - https://github.com/cran
    - This allows anyone to navigate through the code and see the changes made to a package
- For Stata, the **`github`** package, provides an alternative to SSC
    - It can search, install, and manage statistical software for Stata
    - It allows installing previous releases of a statistical package and their dependencies
    - It allows modularizing Stata packages, where other packages can be specified as dependencies and be installed automatically
    - It encourages collaboration on statistical software for Stata

The **github** package for Stata

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

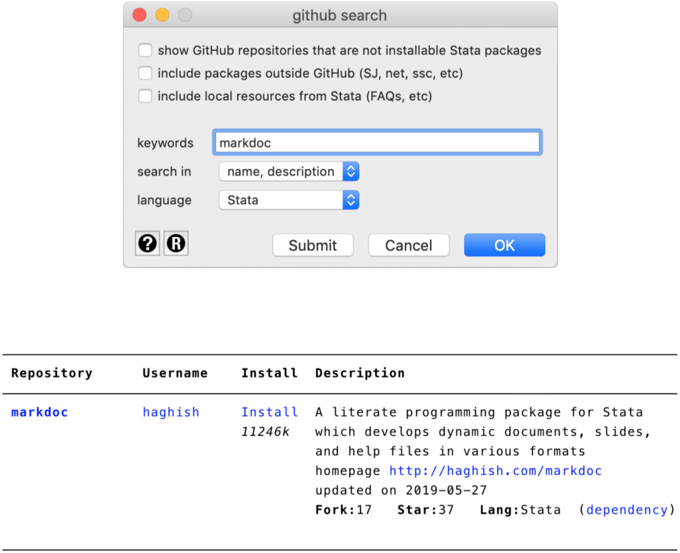
github [*subcommand*] [...] [, options]

Table 1: Summary of **github**'s subcommands

Subcommands	Description
Essential	
`list`	expedites managing packages installed with **github**
`search`	looks for packages or repositories via GitHub API
`install`	installs a package along with its dependencies
`uninstall`	removes a package from Stata
`update`	updates a package to the latest version
Supplementary	
`version`	returns the version of an installed package
`query`	returns all archived stable versions of a package
`check`	tests whether a repository is an installable Stata package

Searching GitHub for Stata packages/repositories

```
. db github
```



github search

- [ ] show GitHub repositories that are not installable Stata packages
- [ ] include packages outside GitHub (SJ, net, ssc, etc)
- [ ] include local resources from Stata (FAQs, etc)

keywords: markdoc

search in: name, description

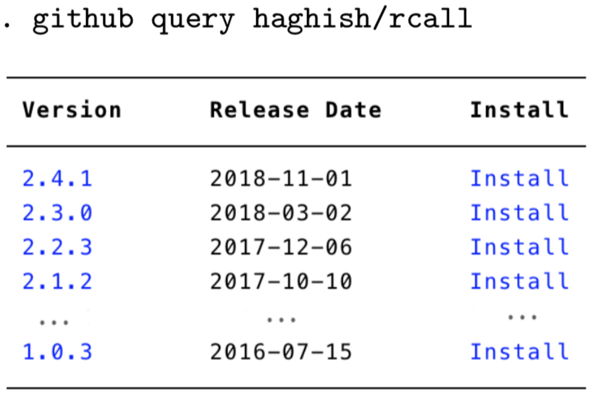
language: Stata

Repository	Username	Install	Description
markdoc	haghish	Install *11246k*	A literate programming package for Stata which develops dynamic documents, slides, and help files in various formats homepage http://haghish.com/markdoc updated on 2019-05-27 **Fork:**17 **Star:**37 **Lang:**Stata (dependency)

Figure 14: Example of searching GitHub for a Stata package

Installing Previous releases of a Stata packages

- The **`github`** package also allows installing older releases
- Software rapidly change and older syntax might not be available in newer releases
- Archiving older releases is necessary to improve reproducibility
  - CRAN archives all versions of a released R package
  - SSC does not archive Stata package versions and only hosts the latest version

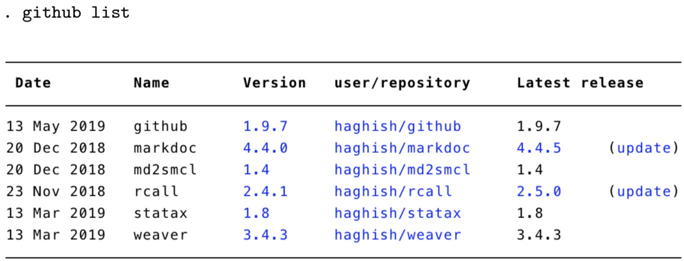


```
. github query haghish/rcall
```

Version	Release Date	Install
2.4.1	2018-11-01	Install
2.3.0	2018-03-02	Install
2.2.3	2017-12-06	Install
2.1.2	2017-10-10	Install
...	...	...
1.0.3	2016-07-15	Install

Managing and updating installed Stata packages

- The **github** package includes commands for managing and updating installed packages
- The **github list** command
  - lists installed packages
  - Current version of the packages
  - Checks whether there is a new release available

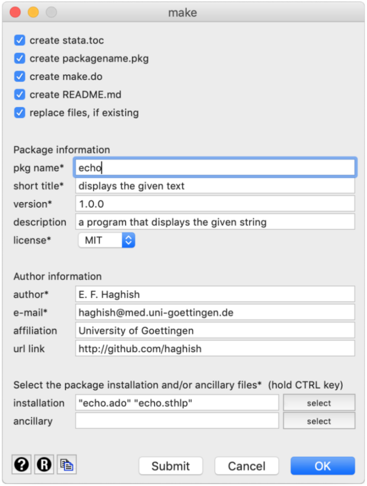


```
. github list

 Date          Name       Version   user/repository     Latest release

 13 May 2019   github     1.9.7     haghish/github      1.9.7
 20 Dec 2018   markdoc    4.4.0     haghish/markdoc     4.4.5      (update)
 20 Dec 2018   md2smcl    1.4       haghish/md2smcl     1.4
 23 Nov 2018   rcall      2.4.1     haghish/rcall       2.5.0      (update)
 13 Mar 2019   statax     1.8       haghish/statax      1.8
 13 Mar 2019   weaver     3.4.3     haghish/weaver      3.4.3
```

Figure 16: Managing and updating installed Stata package

Building a Stata package for GitHub

- The **github** package also includes a command for generating package installation files. The package installation files
  - help the repository to be discovered in search
  - document the creation and update dates as well as the software version



- ✓ create stata.toc
- ✓ create packagename.pkg
- ✓ create make.do
- ✓ create README.md
- ✓ replace files, if existing

Package information

pkg name*	echo
short title*	displays the given text
version*	1.0.0
description	a program that displays the given string
license*	MIT

Author information

author*	E. F. Haghish
e-mail*	haghish@med.uni-goettingen.de
affiliation	University of Goettingen
url link	http://github.com/haghish

Select the package installation and/or ancillary files* (hold CTRL key)

installation	"echo.ado" "echo.sthlp"	select
ancillary		select