Using Github for research and life | **Part II**

Matt Malishev

@darwinanddavis

Applications

Version control

Collaborate

Centralised storage of every possible file type, e.g. Supp Material

Dynamic loading of stored links and programs

```
require(RCurl)
script <- getURL("https://raw.githubusercontent.com/darwina
eval(parse(text = script))
```

Fork and clone a plethora of public data, code, material

Applications

Version control

Collaborate

Centralised storage of every possible file type, e.g. Supp Material

Dynamic loading of stored links and programs

```
require(RCurl)
script <- getURL("https://raw.githubusercontent.com/darwina
eval(parse(text = script))
```

Fork and clone a plethora of public data, code, material

Let's git it

Initialising and using your repo

1. Create a repo

2. Create and stage your files

- add and commit your files

3. Push to a remote github repo

- push your files to your Github

1. Create a repo

Initialise your new local repo

```
# initialise your local git
### <b>
```

Let's git it

Initialising and using your repo

1. Create a repo

2. Create and stage your files

- add and commit your files

3. Push to a remote github repo

- push your files to your Github

1. Create a repo

Initialise your new local repo

```
# initialise your local git
### <b>
```

Let's git it

Initialising and using your repo

1. Create a repo

2. Create and stage your files

- add and commit your files

3. Push to a remote github repo

- push your files to your Github

1. Create a repo

Initialise your new local repo

```
# initialise your local git
### <b>
```

That's it! | Your data is now stored and version controlled in local and remote repos

Cloning an existing repo

Clone a remote repo to your local computer

This creates a git repository on your local machine complete with version control.
Every version of every file for the history of the project is grabbed by default when you run `git clone`.

```
git clone "github url" "new repo name (optional)"
# e.g. git clone https://github.com/darwinanddavis/UsefulCo
```

Why clone?

You can dump the contents of any public repo, including its complete version history, onto your own computer, then upload it onto the cloud.

Workflow example

Open an '.Rproj' with self-contained files

Cloning an existing repo

Clone a remote repo to your local computer

This creates a git repository on your local machine complete with version control.
Every version of every file for the history of the project is grabbed by default when you run `git clone`.

```
git clone "github url" "new repo name (optional)"
# e.g. git clone https://github.com/darwinanddavis/UsefulCo
```

Why clone?

You can dump the contents of any public repo, including its complete version history, onto your own computer, then upload it onto the cloud.

Workflow example

Open an '.Rproj' with self-contained files

Cloning an existing repo

Clone a remote repo to your local computer

This creates a git repository on your local machine complete with version control.
Every version of every file for the history of the project is grabbed by default when you run `git clone`.

```
git clone "github url" "new repo name (optional)"
# e.g. git clone https://github.com/darwinanddavis/UsefulCo
```

Why clone?

You can dump the contents of any public repo, including its complete version history, onto your own computer, then upload it onto the cloud.

Workflow example

Open an '.Rproj' with self-contained files

Cloning an existing repo

Clone a remote repo to your local computer

This creates a git repository on your local machine complete with version control.
Every version of every file for the history of the project is grabbed by default when you run `git clone`.

```
git clone "github url" "new repo name (optional)"
# e.g. git clone https://github.com/darwinanddavis/UsefulCo
```

Why clone?

You can dump the contents of any public repo, including its complete version history, onto your own computer, then upload it onto the cloud.

Workflow example

Open an '.Rproj' with self-contained files

Cloning an existing repo

Clone a remote repo to your local computer

This creates a git repository on your local machine complete with version control.
Every version of every file for the history of the project is grabbed by default when you run `git clone`.

```
git clone "github url" "new repo name (optional)"
# e.g. git clone https://github.com/darwinanddavis/UsefulCo
```

Why clone?

You can dump the contents of any public repo, including its complete version history, onto your own computer, then upload it onto the cloud.

Workflow example

Open an '.Rproj' with self-contained files

Cloning an existing repo

Clone a remote repo to your local computer

This creates a git repository on your local machine complete with version control.
Every version of every file for the history of the project is grabbed by default when you run `git clone`.

```
git clone "github url" "new repo name (optional)"
# e.g. git clone https://github.com/darwinanddavis/UsefulCo
```

Why clone?

You can dump the contents of any public repo, including its complete version history, onto your own computer, then upload it onto the cloud.

Workflow example

Open an '.Rproj' with self-contained files

Cloning an existing repo

Clone a remote repo to your local computer

This creates a git repository on your local machine complete with version control.
Every version of every file for the history of the project is grabbed by default when you run `git clone`.

```
git clone "github url" "new repo name (optional)"
# e.g. git clone https://github.com/darwinanddavis/UsefulCo
```

Why clone?

You can dump the contents of any public repo, including its complete version history, onto your own computer, then upload it onto the cloud.

Workflow example

Open an '.Rproj' with self-contained files

Troubleshooting

Useful `git` syntax

```
git config --global user.name "Matt Malishev"
git config --global user.email "mmlshv@gmail.com"

git diff

git diff # print differences (changes) to files
git diff --stage # show changes in staged gits
```

Extra flags for git commands

```
git log --online # condense log into one line summary
```

Useful `shell` syntax

change working dir to 'Documents'

```
cd ~/Documents
```

move one level up

Troubleshooting

Useful `git` syntax

```
git config --global user.name "Matt Malishev"
git config --global user.email "mmlshv@gmail.com"

git diff

git diff # print differences (changes) to files
git diff --stage # show changes in staged gits
```

Extra flags for git commands

```
git log --online # condense log into one line summary
```

Useful `shell` syntax

change working dir to 'Documents'

```
cd ~/Documents
```

move one level up

Troubleshooting

Useful `git` syntax

```
git config --global user.name "Matt Malishev"
git config --global user.email "mmlshv@gmail.com"

git diff

git diff # print differences (changes) to files
git diff --stage # show changes in staged gits
```

Extra flags for git commands

```
git log --online # condense log into one line summary
```

Useful `shell` syntax

change working dir to 'Documents'

```
cd ~/Documents
```

move one level up

Troubleshooting

Useful `git` syntax

```
git config --global user.name "Matt Malishev"
git config --global user.email "mmlshv@gmail.com"

git diff

git diff # print differences (changes) to files
git diff --stage # show changes in staged gits
```

Extra flags for git commands

```
git log --online # condense log into one line summary
```

Useful `shell` syntax

change working dir to 'Documents'

```
cd ~/Documents
```

move one level up

Troubleshooting

Useful `git` syntax

```
git config --global user.name "Matt Malishev"
git config --global user.email "mmlshv@gmail.com"

git diff

git diff # print differences (changes) to files
git diff --stage # show changes in staged gits
```

Extra flags for git commands

```
git log --online # condense log into one line summary
```

Useful `shell` syntax

change working dir to 'Documents'

```
cd ~/Documents
```

move one level up

Troubleshooting

Useful `git` syntax

```
git config --global user.name "Matt Malishev"
git config --global user.email "mmlshv@gmail.com"

git diff

git diff # print differences (changes) to files
git diff --stage # show changes in staged gits
```

Extra flags for git commands

```
git log --online # condense log into one line summary
```

Useful `shell` syntax

change working dir to 'Documents'

```
cd ~/Documents
```

move one level up