Data Management: Tips and Tools

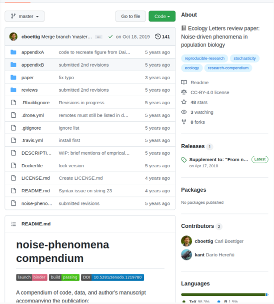
Arnaud Legrand

SMPE lecture

October 2022

A reproducible article

Carl Boettiger: https://github.com/cboettig/noise-phenomena



cboettig Merge branch 'master… ✓ on Oct 18, 2019 141

appendixA	code to recreate figure from Dai…	5 years ago
appendixB	submitted 2nd revisions	5 years ago
paper	fix typo	3 years ago
reviews	submitted 2nd revisions	5 years ago
.Rbuildignore	Revisions in progress	5 years ago
.drone.yml	remotes must still be listed in d…	5 years ago
.gitignore	ignore list	5 years ago
.travis.yml	install first	5 years ago
DESCRIPTI…	WIP: brief mentions of emprical…	5 years ago
Dockerfile	lock version	5 years ago
LICENSE.md	Create LICENSE.md	4 years ago
README.md	Syntax issue on string 23	4 years ago
noise-pheno…	submitted revisions	5 years ago

README.md

noise-phenomena compendium

launch binder | build passing | DOI 10.5281/zenodo.1219780

A compendium of code, data, and author's manuscript accompanying the publication:

About

Ecology Letters review paper: Noise-driven phenomena in population biology

reproducible-research stochasticity ecology research-compendium

Readme
CC-BY-4.0 license
48 stars
3 watching
8 forks

Releases 1

Supplement to: "From n… Latest
on Apr 17, 2018

Packages

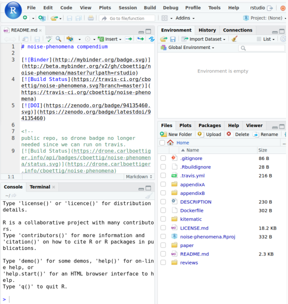
No packages published

Contributors 2

cboettig Carl Boettiger
kant Darío Hereñú

Languages

TeX 98.3% R 1.5%



```
# noise-phenomena compendium

[![Binder](http://mybinder.org/badge.svg)](http://beta.mybinder.org/v2/gh/cboettig/noise-phenomena/master?urlpath=rstudio)
[![Build Status](https://travis-ci.org/cboettig/noise-phenomena.svg?branch=master)](https://travis-ci.org/cboettig/noise-phenomena)
[![DOI](https://zenodo.org/badge/94135460.svg)](https://zenodo.org/badge/latestdoi/94135460)

<!--
public repo, so drone badge no longer needed since we can run on travis.
[![Build Status](https://drone.carlboettiger.info/api/badges/cboettig/noise-phenomena/status.svg)](https://drone.carlboettiger.info/cboettig/noise-phenomena)
```

1:1

```
~/
Type 'license()' or 'licence()' for distribution
details.

R is a collaborative project with many contributo
rs.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in pu
blications.

Type 'demo()' for some demos, 'help()' for on-lin
e help, or
'help.start()' for an HTML browser interface to h
elp.
Type 'q()' to quit R.

>
```

Environment is empty

Name	Size
.gitignore	86 B
.Rbuildignore	28 B
.travis.yml	216 B
appendixA	
appendixB	
DESCRIPTION	230 B
Dockerfile	302 B
kitematic	
LICENSE.md	18.2 KB
noise-phenomena.Rproj	332 B
paper	
README.md	2.3 KB
reviews	

Ingredients:

- ☑ A clean file organization and an `Rmarkdown` document
- ☑ A public `git` repository
- ☑ `Rstudio` in a `Docker` environment available through `Binder`
- ☑ Small non sensitive data set

File Organization and Metadata

Names matter

Courtesy of Data Carpentry and The Turing Way

Bad	Good
myabstract.docx	2014-06-08_abstract-for-sla.docx
Joe’s Filenames Use Spaces and Punctuation.xlsx	joes-filenames-are-getting-better.xlsx
figure 1.png	fig01_scatterplot-talk-length-vs-interest.png
fig 2.png	fig02_histogram-talk-attendance.png
JW7d^(2sl@deletethisandyourcareerisoverWx2*.txt	1986-01-28_raw-data-from-challenger-o-rings.txt

Note: same reason as we have *variable naming conventions*

Bad	Snakecase (C++,(Python, R)	Pascalcase (C#, Go)	Camelcase (Java)
VariAble_1	variable_one	VariableOne	variableOne
variaB1e_two	variable_two	VariableTwo	variableTwo
first_day_of_the_month	day_one	DayOne	dayOne
h	hours_worked	HoursWorked	hoursWorked

Three principles for awesome (file) names

```
.:
01_marshal-data.R         data/                     Makefile
02_pre-dea-filtering.R    helper01_load-counts.R    README
03_dea-wit-limma-voom.R   helper01_load-exp-des.R
04_explore-dea-results.R  LICENCE

./data:
2013-06-26_BRAWNFTEGASSAY_Plasmid-Celline-100-1MutantFraction_A01.csv
2013-06-26_BRAWNFTEGASSAY_Plasmid-Celline-100-1MutantFraction_A02.csv
2013-06-26_BRAWNFTEGASSAY_Plasmid-Celline-100-1MutantFraction_B01.csv
2013-06-26_BRAWNFTEGASSAY_Plasmid-Celline-100-1MutantFraction_B02.csv
```

Plays well with default ordering

- Numeric first
- YYYY-MM-DD for dates (ISO 8601)
- Left pad numbers with 0.

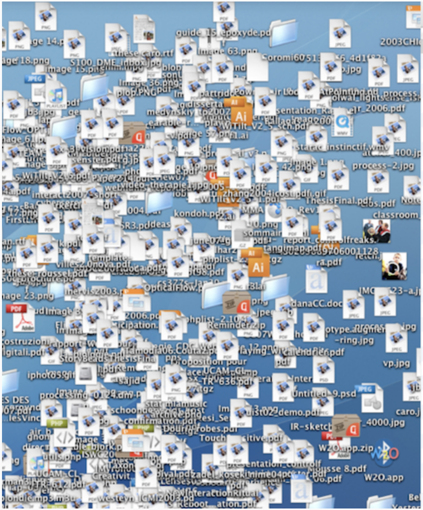
Human readable Name contains info (meta-data) on content

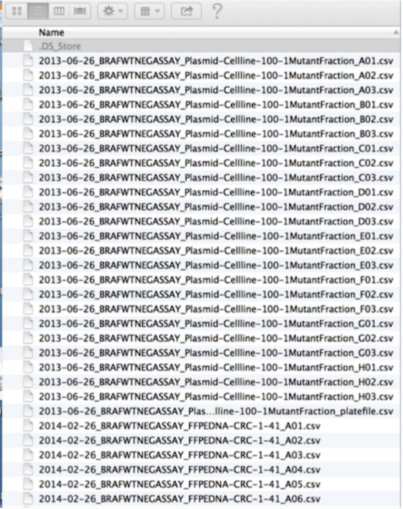
Machine readable

- Regular expression and globbing friendly
  - Avoid spaces, punctuation, accented characters, case sensitivity
  - Easy to compute on
- Deliberate and consistent use of delimiters (_ and -)

Organization

Organization





Name

.DS_Store
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_A01.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_A02.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_A03.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_B01.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_B02.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_B03.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_C01.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_C02.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_C03.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_D01.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_D02.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_D03.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_E01.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_E02.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_E03.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_F01.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_F02.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_F03.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_G01.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_G02.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_G03.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_H01.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_H02.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_H03.csv
2013-06-26_BRAFWTNEGASSAY_Plas...lline-100-1MutantFraction_platefile.csv
2014-02-26_BRAFWTNEGASSAY_FFPEDNA-CRC-1-41_A01.csv
2014-02-26_BRAFWTNEGASSAY_FFPEDNA-CRC-1-41_A02.csv
2014-02-26_BRAFWTNEGASSAY_FFPEDNA-CRC-1-41_A03.csv
2014-02-26_BRAFWTNEGASSAY_FFPEDNA-CRC-1-41_A04.csv
2014-02-26_BRAFWTNEGASSAY_FFPEDNA-CRC-1-41_A05.csv
2014-02-26_BRAFWTNEGASSAY_FFPEDNA-CRC-1-41_A06.csv

Organization

Face it...

- There are going to be files LOTS of files
  - Raw data, Ready to analyze data, computational results,
  - Figures, tables
  - Reports, manuscripts, slides, posters
- The files will change over time
- The files will have relationships to each other
- It'll probably get complicated

Mighty weapon

- File organization and naming is a mighty weapon against chaos
- Make a file's name and location VERY INFORMATIVE about
  - What it is, why it exists how it relates to other things
- READMEs are great, but the more things are self-explanatory, the better

General tips

- Keep all files associated with a project in a single folder
    - Different projects should have separate folders
    - Version control everything with git
    - Separate public/private/secret ? Separate by folder (and Git repo)

General tips

- Keep all files associated with a project in a single folder
    - Different projects should have separate folders
    - Version control everything with git
    - Separate public/private/secret ? Separate by folder (and Git repo)
- All data in `data/`
    - `raw_data/from_alice`, `raw_data/from_bob`
    - `derived_data/from_alice`, `derived_data/from_bob`,

General tips

- Keep all files associated with a project in a single folder
    - Different projects should have separate folders
    - Version control everything with git
    - Separate public/private/secret ? Separate by folder (and Git repo)
- All data in `data/`
    - `raw_data/from_alice`, `raw_data/from_bob`
    - `derived_data/from_alice`, `derived_data/from_bob`,
- All code in the `src/`, `source/`, `code/` directory (**Pick one!**)
    - When software is reused in several projects it can make sense to put them in own repo (maybe use `git submodule`).
    - Describe your software dependencies (`requirements.txt,` `Dockerfile`, ...)

General tips

- Keep all files associated with a project in a single folder
  - Different projects should have separate folders
  - Version control everything with git
  - Separate public/private/secret ? Separate by folder (and Git repo)
- All data in `data/`
  - `raw_data/from_alice`, `raw_data/from_bob`
  - `derived_data/from_alice`, `derived_data/from_bob`,
- All code in the `src/`, `source/`, `code/` directory (**Pick one!**)
  - When software is reused in several projects it can make sense to put them in own repo (maybe use `git submodule`).
  - Describe your software dependencies (`requirements.txt,` `Dockerfile`, …)
- Add a `README` file to describe the project and instructions on reproducing the results
  - Talk to others in the project about what you do and write it down

General tips

- Keep all files associated with a project in a single folder
  - Different projects should have separate folders
  - Version control everything with git
  - Separate public/private/secret ? Separate by folder (and Git repo)
- All data in `data/`
  - `raw_data/from_alice`, `raw_data/from_bob`
  - `derived_data/from_alice`, `derived_data/from_bob`,
- All code in the `src/`, `source/`, `code/` directory (**Pick one!**)
  - When software is reused in several projects it can make sense to put them in own repo (maybe use `git submodule`).
  - Describe your software dependencies (`requirements.txt,` `Dockerfile`, …)
- Add a `README` file to describe the project and instructions on reproducing the results
  - Talk to others in the project about what you do and write it down
- Add an `AUTHOR`, `CONTRIBUTING`, `CODE_OF_CONDUCT` file

General tips

- Keep all files associated with a project in a single folder
  - Different projects should have separate folders
  - Version control everything with git
  - Separate public/private/secret ? Separate by folder (and Git repo)
- All data in `data/`
  - `raw_data/from_alice`, `raw_data/from_bob`
  - `derived_data/from_alice`, `derived_data/from_bob`,
- All code in the `src/`, `source/`, `code/` directory (**Pick one!**)
  - When software is reused in several projects it can make sense to put them in own repo (maybe use `git submodule`).
  - Describe your software dependencies (`requirements.txt,` `Dockerfile`, ...)
- Add a `README` file to describe the project and instructions on reproducing the results
  - Talk to others in the project about what you do and write it down
- Add an `AUTHOR`, `CONTRIBUTING`, `CODE_OF_CONDUCT` file
- Include appropriate `LICENSE` file and information on software requirements

Many possible file organizations

- The Turing Way

```
project_folder/
├── docs                        # documentation
│   └── codelist.txt
│   └── project_plan.txt
│   └── ...
│   └── deliverables.txt
├── data
│   └── raw/
│       └── my_data.csv
│   └── clean/
│       └── data_clean.csv
├── analysis                    # scripts
│   └── my_script.R
├── results                     # analysis output
│   └── figures
├── .gitignore                  # files excluded from git vers
├── install.R                   # environment setup
├── CODE_OF_CONDUCT             # Code of Conduct for communit
├── CONTRIBUTING                # Contribution guideline for c
├── LICENSE                     # software license
├── README.md                   # information about the repo
└── report.md                   # report of project
```

Pick a strategy, any strategy, just pick one and stick to it!

Many possible file organizations

- The Turing Way
- CodeRefinery’s suggestions

```
project_name/
├── README.md             # overview of the project
├── data/                 # data files used in the project
│   ├── README.md         # describes where data came from
│   └── subfolder/        # may contain subdirectories
├── processed_data/       # intermediate files from the ana
├── manuscript/           # manuscript describing the resul
├── results/              # results of the analysis (data,
├── src/                  # contains all code in the projec
│   ├── LICENSE           # license for your code
│   ├── requirements.txt  # software requirements and depen
│   └── ...
└── doc/                  # documentation for your project
    ├── index.rst
    └── ...
```

Pick a strategy, any strategy, just pick one and stick to it!

Many possible file organizations

- The Turing Way
- CodeRefinery’s suggestions
- Ben Marwick’s R compendium (rrtools)

```
Dockerfile
R/                          # R scripts
analysis/
|
├── paper/
│   ├── paper.Rmd       # this is the main document to edit
│   └── references.bib  # this contains the reference list
├── figures/            # location of the figures produced
|
├── data/
│   ├── raw_data/       # data obtained from elsewhere
│   └── derived_data/   # data generated during the analysi
|
└── templates
    ├── journal-of-archaeological-science.csl
    |                   # this sets the style of citations
    ├── template.docx   # used to style the output of the p
    └── template.Rmd
```

Pick a strategy, any strategy, just pick one and stick to it!

Many possible file organizations

- The Turing Way
- CodeRefinery's suggestions
- Ben Marwick's R compendium (rrtools)
- Cookiecutter (e.g., a Snakemake template)

```
project_name/
├── .gitignore
├── README.md
├── LICENSE.md
├── config.yaml
├── scripts
|     ├── script1.py
|     └── script2.R
├── envs
|      └── myenv.yaml
└── Snakefile
```

Pick a strategy, any strategy, just pick one and stick to it!

Git and Git Annex

Git

- Designed by Linus Torvald in 2005 (BitKeeper licensing issues)
- Allows to track versions (i.e., to manage an history) in a distributed way (Introduction to Git without the command line)
- Although many common git workflows are centralized (e.g., through github and gitlab), git is ditributed

Main drawback: git has been designed and optimized to handle for source code, not large binary files

Christmas List for Data Management

1. A lightweight `git clone` (do not necessarily download all large files)
    - I.e., more than git tricks (`git clone --depth` and `git subtree`)
2. Garbage collection, i.e., allows to delete large files (even in `.git/`)
3. Get large files on demand and guarantee to get the right ones
4. Allow handling different (possibly unreliable) storage media

Christmas List for Data Management

1. A lightweight `git clone` (do not necessarily download all large files)
    - I.e., more than git tricks (`git clone --depth` and `git subtree`)
2. Garbage collection, i.e., allows to delete large files (even in `.git/`)
3. Get large files on demand and guarantee to get the right ones
4. Allow handling different (possibly unreliable) storage media

Several proposed extensions for handling large files:

Git LFS

- Centralized and supported by GitHub (hence by GitLab)
- Easy to use but fails all previous requirements

Git Annex by Joey Hess (Debian, Haskell)

- Steeper learning curve but incredibly powerful

Git Annex Principles

- The project is populated with symbolic links to the large files which end up in `.git/annex/objects`

```
data/raw_data/uset/Wlight/2021/06/UPH20210610112235.FTS ->
../../../../../../.git/annex/objects/fw/j8/SHA256E-s8392320--d59d841adb2f5f9eb30d115
s8392320--d59d841adb2f5f9eb30d11501440ce53539bcb9aec95b80f6877d2169e8c6481.FTS
```

- You may `git annex drop` files (remove from the annex)
- Large files are generally identified by their content (SHA256)
  ⇝ Check content when `git annex get`
- Remotes are ways to access files (a USB key, a server through SSH or webdav, a web server, Amazon S3, etc.)
  - Files may be duplicated/migrated between remotes
- Information on the remotes is stored in a special `git-annex` branch which must be synchronized between git repos

Demo: Initialization

```
pwd
git init
echo "Hello" > README
git add README ; git commit -m "Initial commit"
git branch
git annex init "My laptop"
git branch
```

```
/tmp/test-git-annex
Initialized empty Git repository in /tmp/test-git-annex/.git/
icarus:/tmp/test-git-annex$ [master (root-commit) 7f50a1f] Initial commit
 1 file changed, 1 insertion(+)
 create mode 100644 README
* master
init My laptop ok
(recording state in git...)
git-annex
* master
```

Demo: Adding data

```
mkdir -p data/
git config annex.largefiles 'largerthan=100kb and include=data/*'
```

```
echo "random; stuff" > data/foo.csv
dd if=/dev/zero of=data/zero.dat bs=1M count=1
ls -l data/
git annex add data/* ## should be git add !!
ls -l data/
git commit -m "Adding data files"
```

```
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00545621 s, 192 MB/s
-rw-r--r-- 1 alegrand alegrand      14 Oct 26 23:15 foo.csv
-rw-r--r-- 1 alegrand alegrand 1048576 Oct 26 23:15 zero.dat
add data/foo.csv (non-large file; adding content to git repository) ok
add data/zero.dat ok
(recording state in git...)
-rw-r--r-- 1 alegrand alegrand  14 Oct 26 23:15 foo.csv
../.git/annex/objects/fP/jz/SHA256E-s1048576--30e14955ebf1352266dc2ff8067e681046
s1048576--30e14955ebf1352266dc2ff8067e68104607e750abb9d3b36582b8af909fcb58.dat
[master 91c2449] Adding data files
 2 files changed, 2 insertions(+)
 create mode 100644 data/foo.csv
 create mode 120000 data/zero.dat
```

Demo: Adding a remote on a USB key (1/2)

```
rm -rf /media/alegrand/7C78-3F81/test-git-annex
```

```
cd /media/alegrand/7C78-3F81
git clone /tmp/test-git-annex
ls -lR test-git-annex/
```

```
Cloning into 'test-git-annex'...
done.
test-git-annex/:
total 8
drwxr-xr-x 2 alegrand alegrand 4096 Oct 26 23:16 data
-rw-r--r-- 1 alegrand alegrand    6 Oct 26 23:16 README

test-git-annex/data:
total 8
-rw-r--r-- 1 alegrand alegrand  14 Oct 26 23:16 foo.csv
-rw-r--r-- 1 alegrand alegrand 201 Oct 26 23:16 zero.dat
```

Demo: Adding a remote on a USB key (2/2)

Let's initialize the USB key and tell it about the laptop

```
cd test-git-annex
git annex init "portable USB drive"
git remote add laptop /tmp/test-git-annex
```

```
init portable USB drive
  Detected a filesystem without fifo support.
  Disabling ssh connection caching.
  Detected a crippled filesystem.
  Entering an adjusted branch where files are unlocked as this filesystem does n
Switched to branch 'adjusted/master(unlocked)'
ok
(recording state in git...)
```

Let's tell the laptop about the USB key

```
cd /tmp/test-git-annex
git remote add usbdrive /media/alegrand/7C78-3F81/test-git-annex
```

Demo: Syncing the annexes (not the content!)

```
cd /media/alegrand/7C78-3F81/test-git-annex
git annex sync laptop
```

```
commit
On branch adjusted/master(unlocked)
nothing to commit, working tree clean
ok
pull laptop
From /tmp/test-git-annex
laptop/git-annex
laptop/master
ok
push laptop
Enumerating objects: 5, done.
Delta compression using up to 4 threads
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To /tmp/test-git-annex
synced/master
synced/git-annex
ok
```

Demo: Getting file content

Now get the content!

```
cd /media/alegrand/7C78-3F81/test-git-annex
git annex get data/zero.dat
```

```
get data/zero.dat (from laptop...)
31.98 KiB       268 KiB/s 3s
9%    95.95 KiB       223 KiB/s 4s
16%   159.92 KiB      228 KiB/s 3s
22%   223.89 KiB      229 KiB/s 3s
28%   287.86 KiB      219 KiB/s 3s
34%   351.83 KiB      228 KiB/s 2s
41%   415.8 KiB       228 KiB/s 2s
47%   479.77 KiB      229 KiB/s 2s
53%   543.73 KiB      232 KiB/s 2s
59%   607.7 KiB       232 KiB/s 1s
66%   671.67 KiB      232 KiB/s 1s
72%   735.64 KiB      228 KiB/s 1s
78%   799.61 KiB      229 KiB/s 0s
84%   863.58 KiB      229 KiB/s 0s
91%   927.55 KiB      229 KiB/s 0s
97%   991.52 KiB      231 KiB/s 0s
100%  1 MiB           117 KiB/s 0s
```

Demo: Dropping files

Let's try to get rid of the big file on my laptop

```
cd /tmp/test-git-annex
git annex drop data/zero.dat
```

```
drop data/zero.dat (unsafe)
  Could only verify the existence of 0 out of 1 necessary copy
  Rather than dropping this file, try using: git annex move
  (Use --force to override this check, or adjust numcopies.)
failed
drop: 1 failed
```

Demo: Dropping files (2/2)

```
git annex sync
git annex drop data/zero.dat
```

```
commit
On branch master
nothing to commit, working tree clean
ok
pull usbdrive
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 9 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (9/9), 852 bytes | 106.00 KiB/s, done.
From /media/alegrand/7C78-3F81/test-git-annex
 * [new branch]      adjusted/master(unlocked) -> usbdrive/adjusted/master(unloc
 * [new branch]      git-annex                 -> usbdrive/git-annex
 * [new branch]      master                    -> usbdrive/master
 * [new branch]      synced/master             -> usbdrive/synced/master
ok
(merging usbdrive/git-annex into git-annex...)
drop data/zero.dat ok
```

Using `git annex move --to usbdrive` in the first place would have been more convenient.

Demo: Getting files back (2/2)

```
git annex get data/zero.dat
```

```
get data/zero.dat (from usbdrive...)
ok
(recording state in git...)
```

Large files cannot be modified inadvertently

```
dd if=/dev/zero of=data/zero.dat bs=2M count=1
```

```
dd: failed to open 'data/zero.dat': Permission denied
```

You should `git annex unlock` them first.

```
dd if=/dev/zero of=data/zero2.dat bs=2M count=1
git annex add data/zero2.dat
git annex move data/zero2.dat --to usbdrive
```

```
1+0 records in
1+0 records out
2097152 bytes (2.1 MB, 2.0 MiB) copied, 0.0106413 s, 197 MB/s
add data/zero2.dat
31.98 KiB        14 MiB/s 0s
100%  2 MiB           110 MiB/s 0s

ok
(recording state in git...)
move data/zero2.dat (to usbdrive...)
ok
(recording state in git...)
```

There is more!

```
cd data/
git annex addurl --preserve-filename --pathdepth=2 \
    https://www.sidc.be/DATA/uset/Wlight/2014/06/UPH20140601105039.FTS
```

```
addurl https://www.sidc.be/DATA/uset/Wlight/2014/06/UPH20140601105039.FTS
(to uset/Wlight/2014/06/UPH20140601105039.FTS) ok
(recording state in git...)
```

`git-annex` can also store files in Amazon S3, Glacier, on a rsync server, in WebDAV, or even pull files down from the web and bittorrent.

Bonus: Files stored on special remotes can easily be encrypted!

Archiving

Archives

or = awesome collaborations (≠ archive)

- D. Spinellis. *The Decay and Failures of URL References*. CACM, 46(1), 2003
  The half-life of a referenced URL is approximately 4 years from its publication date.
- P. Habibzadeh. *Decay of References to Web sites in Articles Published in General Medical Journals: Mainstream vs Small Journals*. Applied Clinical Informatics. 4 (4), 2013
  half life ranged from 2.2 years in EMHJ to 5.3 years in BMJ
- Discontinuated forges: Code Space, Gitorious, Google code, Inria Gforge

Archives

or = awesome collaborations (≠ archive)

- D. Spinellis. *The Decay and Failures of URL References*. CACM, 46(1), 2003
  The half-life of a referenced URL is approximately 4 years from its publication date.
- P. Habibzadeh. *Decay of References to Web sites in Articles Published in General Medical Journals: Mainstream vs Small Journals*. Applied Clinical Informatics. 4 (4), 2013
  half life ranged from 2.2 years in EMHJ to 5.3 years in BMJ
- Discontinuated forges: Code Space, Gitorious, Google code, Inria Gforge

Article archives arXiv.org HAL archives-ouvertes.fr

Data archives figshare zenodo

Software Archive Software Heritage Collect/Preserve/Share

INTERNET ARCHIVE

Archiving from Github on Zenodo

Zenodo was created by OpenAIRE and CERN to provide a place for researchers to deposit datasets. It was launched in 2013, allowing researchers from any domain to upload files up to 50 GB.

Zenodo has a special integration with GitHub to make code hosted in GitHub easy to cite and archive.

Once configured, each time you create a new GitHub release:

- Github creates a `zip` file of the head of your repository
- Uploads it on Zenodo
- Zenodo issues a new DOI

Remember Carl Boettiger's reproducible article ?

This will obviously not work with `git annex` nor `git lfs` (see https://zenodo.org/record/6361006#.Y1mt29JBw1u) but there is a prototype.

Containers and package managers

WIP slides

- seq1-sw_env_intro~unit1-lecture~slides.pdf
- seq2-package_mgmt~unit1-lecture~slides.pdf
- seq3-isolation_and_containers~unit1-lecture~slides.pdf
- seq3-isolation_and_containers~unit2-lecture~slides.pdf