DROPPING AND REMOVING STUFF

What to do with files you don't want to keep

`datalad drop` and `datalad remove`

Code: psychoinformatics-de.github.io/rdm-course/92-filesystem-operations

DROP & REMOVE

- Try to remove (*rm*) one of the pictures in your dataset. What happens?
- Version control tools keep a revision history of your files - file contents are not actually removed when you *rm* them. Interactions with the revision history of the dataset can bring them "back to life"

DROP & REMOVE

- Clone a small example dataset to drop file contents and remove datasets:

```
$ datalad clone https://github.com/datalad-datasets/machinelearning-books.git
$ cd machinelearning-books
$ datalad get A.Shashua-Introduction_to_Machine_Learning.pdf
```

- **datalad drop** removes annexed file contents from a local dataset annex and frees up disk space. It is the antagonist of **get** (which can get files and subdatasets).

```
$ datalad drop A.Shashua-Introduction_to_Machine_Learning.pdf
drop(ok): /tmp/machinelearning-books/A.Shashua-Introduction_to_Machine_Learning.pdf (file)
          [checking https://arxiv.org/pdf/0904.3664v1.pdf...]
```

- But: Default safety checks require that dropped files can be re-obtained to prevent accidental data loss. **git annex whereis** reports all registered locations of a file's content
- **drop** does not only operate on individual annexed files, but also directories, or globs, and it can uninstall subdatasets:

```
$ datalad clone https://github.com/datalad-datasets/human-connectome-project-openaccess.git
$ cd human-connectome-project-openaccess
$ datalad get -n HCP1200/996782
$ datalad drop --what all  HCP1200/996782
```

DROP & REMOVE

- **datalad remove** removes complete dataset or dataset hierarchies and leaves no trace of them. It is the antagonist to **clone**.

```
# The command operates outside of the to-be-removed dataset!
$ datalad remove -d . machinelearning-books
uninstall(ok): /tmp/machinelearning-books (dataset)
```

- But: Default safety checks require that it could be re-cloned in its most recent version from other places, i.e., that there is a *sibling* that has all revisions that exist locally **datalad siblings** reports all registered siblings of a dataset.

DROP & REMOVE

- Create a dataset from scratch and add a file

```
$ datalad create local-dataset
$ cd local-dataset
$ echo "This file content will only exist locally" > local-file.txt
$ datalad save -m "Added a file without remote content availability"
```

- **datalad drop** refuses to remove annexed file contents if it can't verify that **datalad get** could re-retrieve it

```
$ datalad drop local-file.txt
$ drop(error): local-file.txt (file) [unsafe; Could only verify the existence of 0 out of 1 necessary copy
              (Note that these git remotes have annex-ignore set: origin upstream);
              (Use --reckless availability to override this check, or adjust numcopies.)]
```

- Adding **--reckless availability** overrides this check

```
$ datalad drop local-file.txt --reckless availability
```

- Be mindful that **drop** will only operate on the most recent version of a file - past versions may still exist afterwards unless you drop them specifically. **git annex unused** can identify all files that are left behind

DROP & REMOVE

- **datalad remove** refuses to remove datasets without an up-to-date *sibling*

```
$ datalad remove -d local-dataset
uninstall(error): . (dataset) [to-be-dropped dataset has revisions that are not available at
                   sibling. Use `datalad push --to ...` to push these before dropping the loca
                   or ignore via `--reckless availability`. Unique revisions: ['main']]
```

- Adding **--reckless availability** overrides this check

```
$ datalad remove -d local-dataset --reckless availability
```

REMOVING WRONGLY

- Using a file browser or command line calls like **rm -rf** on datasets is doomed to fail. Recreate the local dataset we just removed:

```
$ datalad create local-dataset
$ cd local-dataset
$ echo "This file content will only exist locally" > local-file.txt
$ datalad save -m "Added a file without remote content availability"
```

- Removing it the wrong way causes chaos and leaves an usuable dataset corpse behind:

```
$ rm -rf local-dataset
rm: cannot remove 'local-dataset/.git/annex/objects/Kj/44/MD5E-s42--8f008874ab52d0ff02a5bbd0174ac95e.txt/
MD5E-s42--8f008874ab52d0ff02a5bbd0174ac95e.txt': Permission denied
```

- The dataset can't be fixed, but to remove the corpse **chmod** (change file mode bits) it (i.e., make it writable)

```
$ chmod +w -R local-dataset
$ rm -rf local-dataset
```

QUESTIONS!

Awkward silence can be bridged with awkward MC questions :)

http://etc.ch/XcWM

Direct Poll