

Best Practices in Handling Genomic Data

or

Managing the Storage and Analysis of ...

Douglas G Scofield

Evolutionary Biology Centre & UPPMAX, Uppsala University

douglas.scofield@ebc.uu.se

douglasgscofield@gmail.com

<https://www.dropbox.com/s/sqtap845a10fnhi/Scofield.pdf?dl=1>

About me: first, Computers

- BS, Computer Science, Michigan State, 1988
- Software engineer, 9 years, compiler internals
- but... south Florida, subtropical, 25°-26°N

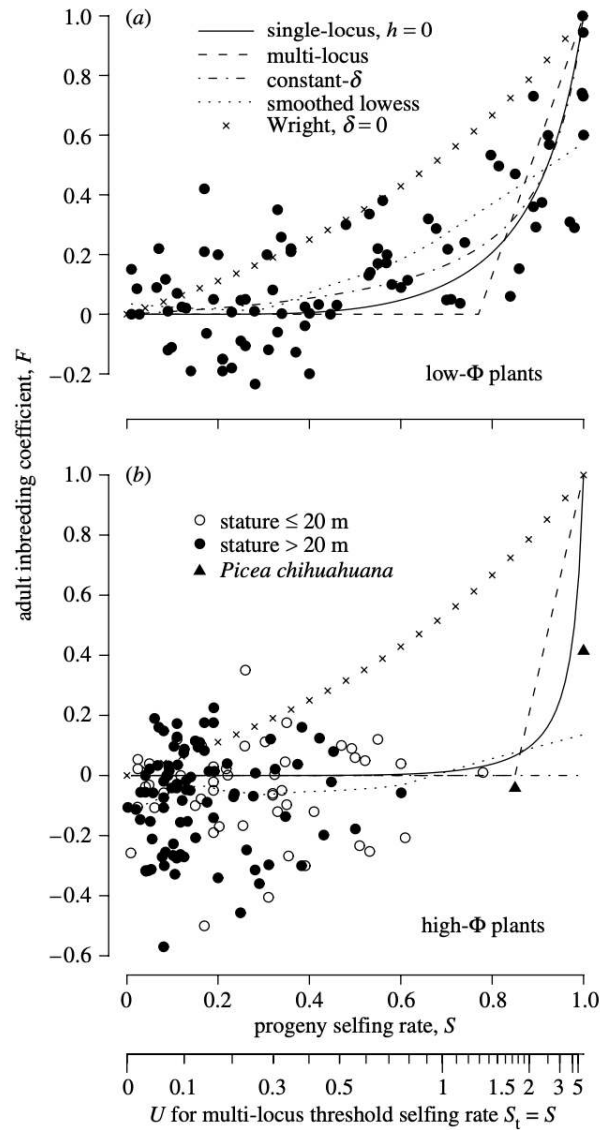


Computers, then plants

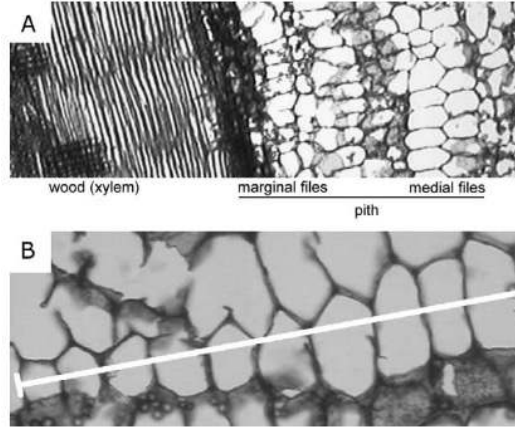
- BS, Computer Science, Michigan State, 1988
- Software engineer, 9 years, compiler internals
- BS, Botany, Florida Atlantic Univ, 1997
- PhD, Biology, University of Miami, 2004



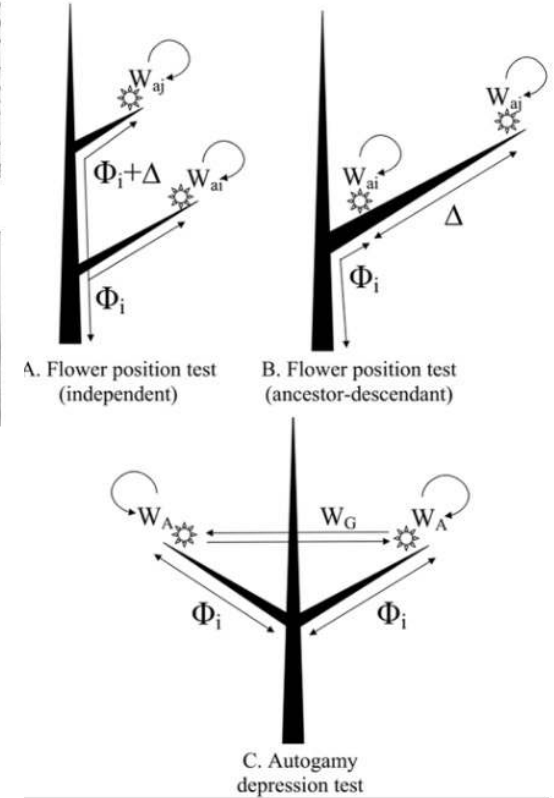
Evolutionary consequences of plant stature



Scofield & Schultz *PRSLB* 2006



Scofield *Am J Bot* 2006



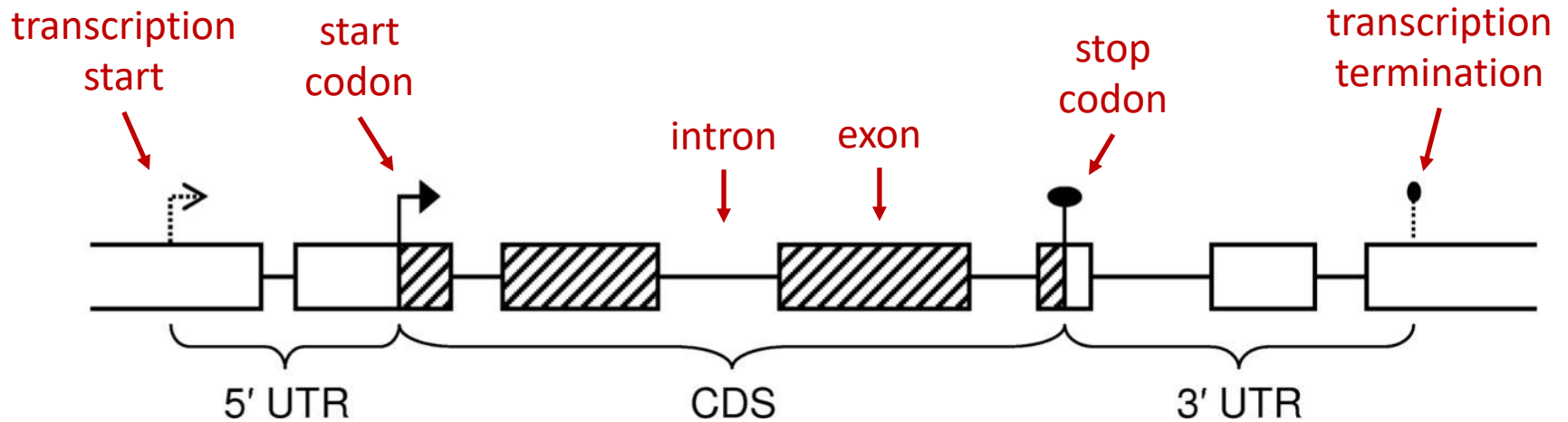
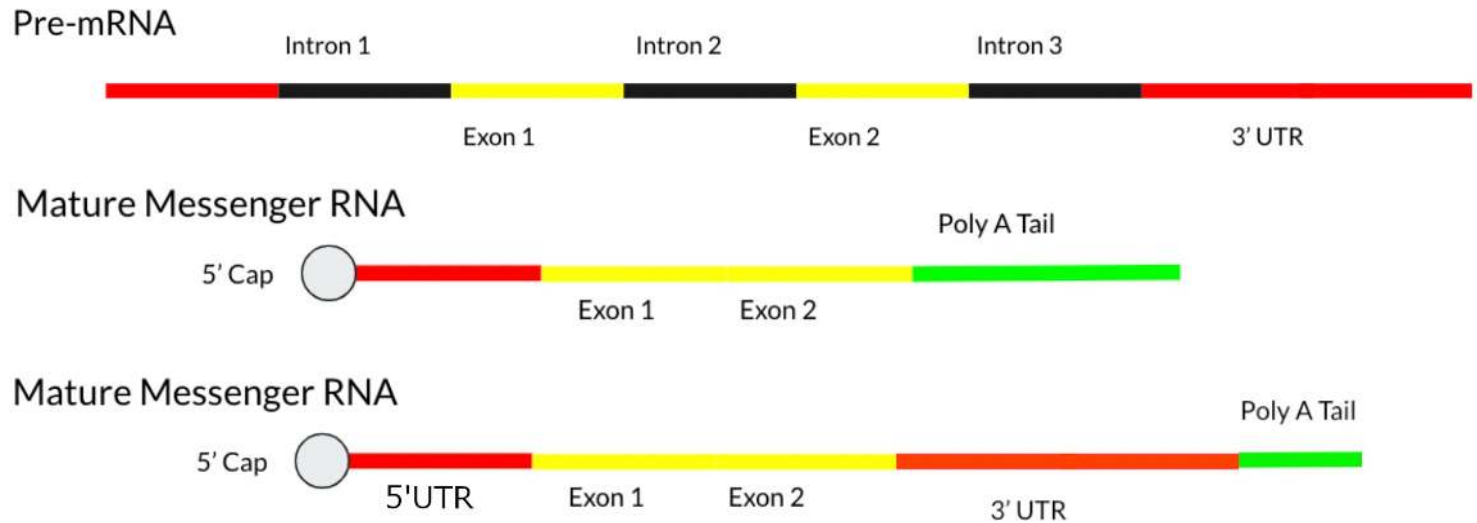
Schultz & Scofield *Am Nat* 2009

Computers, then plants, then genomics

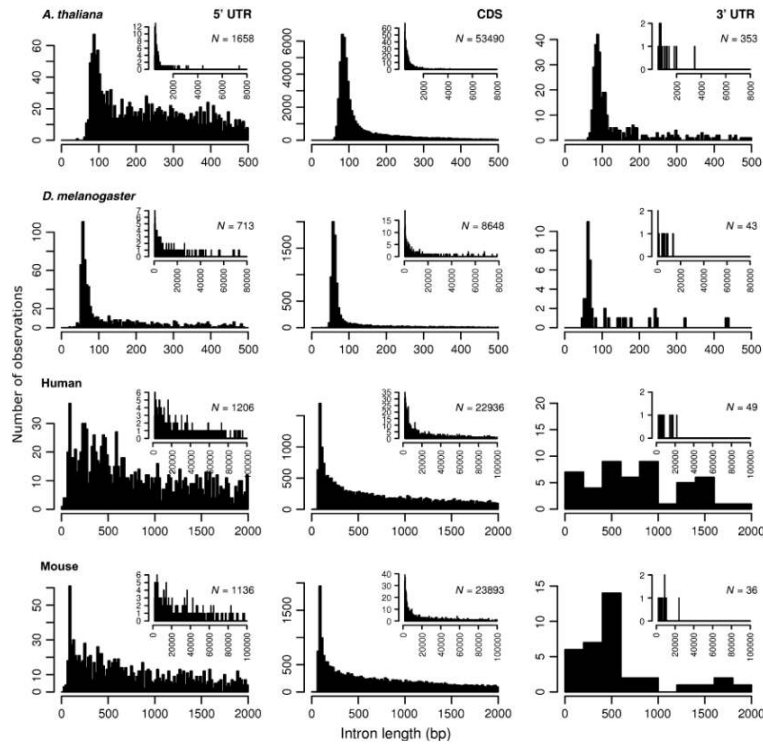
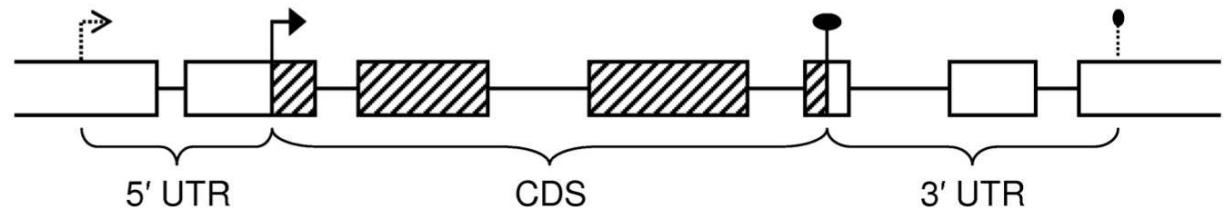
- BS, Computer Science, Michigan State, 1988
- Software engineer, 9 years, compiler internals
- BS, Botany, Florida Atlantic Univ, 1997
- PhD, Biology, University of Miami, 2004
- Post-doc, Michael Lynch, Indiana Univ, 2004-2007



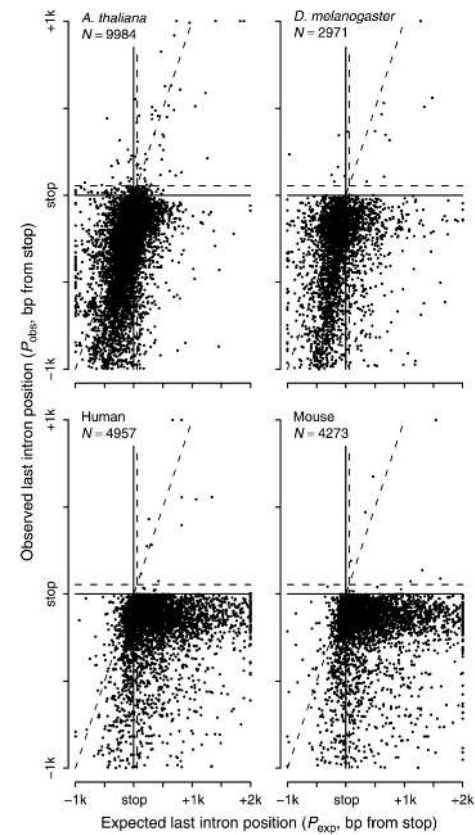
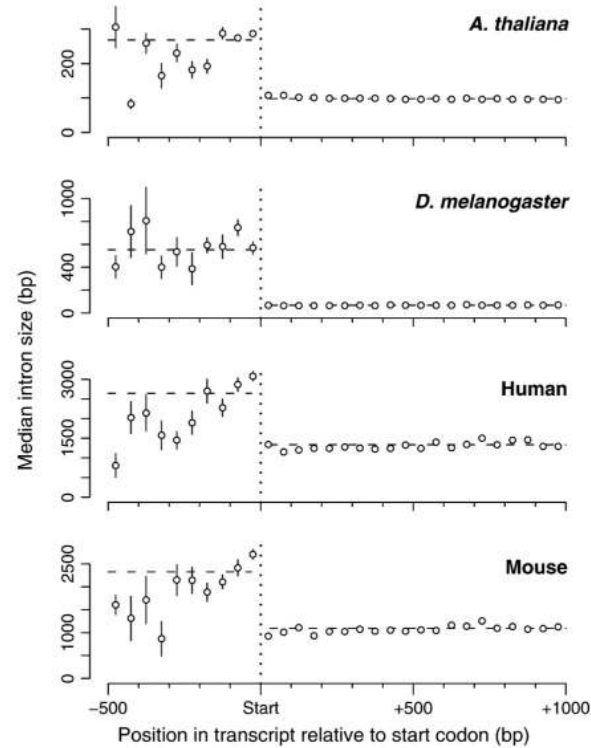
Gene structure: introns in UTRs



Gene structure: introns in UTRs



Hong, Scofield, Lynch *Mol Biol Evol* 2006



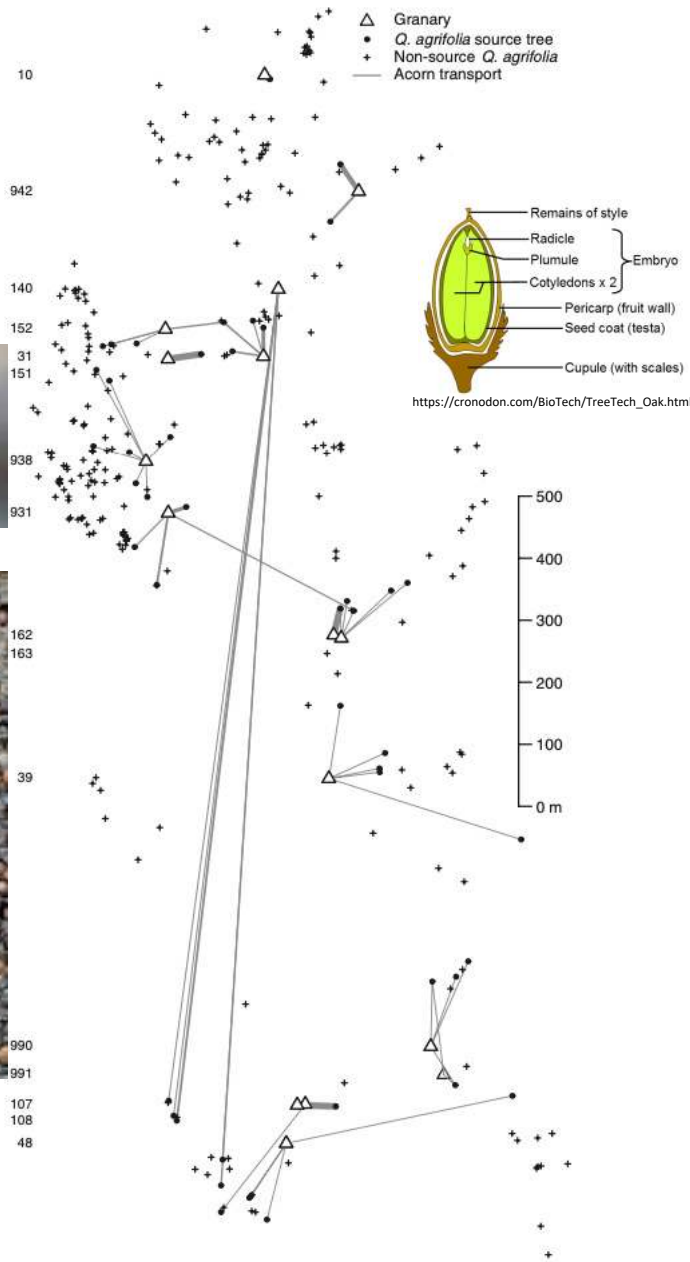
Scofield *Mol Biol Evol* 2007

Computers, then plants, then genomics

- BS, Computer Science, Michigan State, 1988
- Software engineer, 9 years, compiler internals
- BS, Botany, Florida Atlantic Univ, 1997
- PhD, Biology, University of Miami, 2004
- Post-doc, Michael Lynch, Indiana Univ, 2004-2007
- Post-doc, Victoria Sork, UCLA, 2007-2010

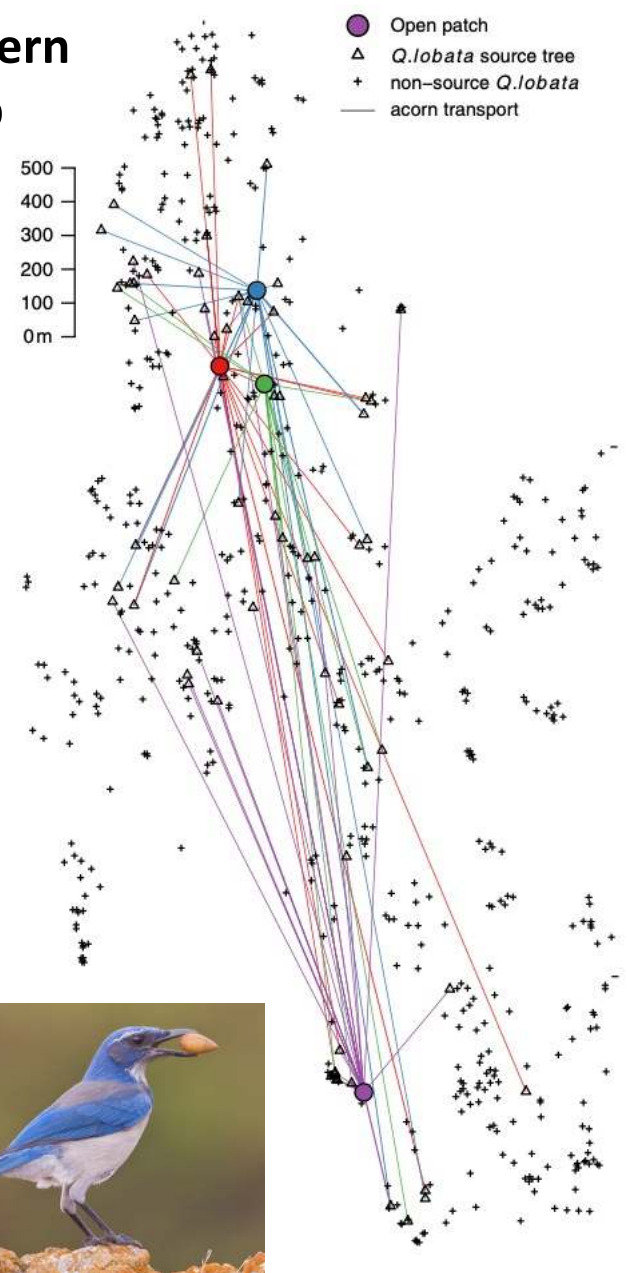


Acorn woodpeckers



Scofield, Sork, Smouse, *J Ecol* 2010

Western scrub jay



Sork et al *Evol Ecol* 2015

Computers, then plants, then genomics

- BS, Computer Science, Michigan State, 1988
- Software engineer, 9 years, compiler internals
- BS, Botany, Florida Atlantic Univ, 1997
- PhD, Biology, University of Miami, 2004
- Post-doc, Michael Lynch, Indiana Univ, 2004-2007
- Post-doc, Victoria Sork, UCLA, 2007-2010
- Research engineer, McGill University 2010-11
- Research engineer, UPSC, Umeå Univ 2011-13
- Research bioinformatician + Application expert, Uppsala University 2013-present



Outline

- Assumptions
- Off-site backup
- Archives, compression and checksums
- Managing project access with Linux groups
- Raw data and metadata
- Organizing raw data, and creating *views*
- Organizing the rest of the project: the fun stuff!
- Shell job control, and running within screen

Some things not covered

- Source/version control
 - git
- Scripting (just a very tiny bit)
- Sysadmin tasks
 - creating groups, adding users to groups, sudo
- Batch system details
 - SLURM, SGE, etc
 - Resources required to run particular tasks
- “Wars”
 - which editor ... which scripting language ... which shell ...
 - Nearly neutral vs fluctuating selection ... splitters vs lumpers ...

Compute resource assumptions

- Linux, or
- Mac, with Homebrew or MacPorts
- Windows, with some form of Linux emulation
- A "large amount" of attached storage is under your control
 - How large? 3-5 times the size of your raw data, and more is better
 - Backup of this 'working' storage is not required ... BUT
- Off-site backup is available for essentials
 - How large? Perhaps 1.5 times the size of your raw data
 - Raw data + essential resources: scripts, important intermediate results, final results, and so on

Regulatory and conduct assumptions

- Raw data and accompanying metadata must be backed up

Metadata: data about data

- Lab notebooks and work logs are kept
- Data will be shared
- Methods will be shared
- Your work must be reproducible

"Off-site" backup ?

- Backup that is not directly attached to your working storage
or
- Storage that is itself backed up off-site
- Data types may be split between different off-site backups
 - raw data in one
 - scripts in another (e.g., Github, Gitlab, ...)
 - important intermediate and final results in another
- Back up changing data types regularly
 - new raw data
 - scripts and other results, best when combined with *version control*

Off-site backup for raw data

- Large size
- Low frequency of access
 - Upload whenever there is new data
 - Download (hopefully!) rarely
- Version control not required
- Block storage, like Linux filesystems
 - each file chunked into blocks, random access to any part of a file

or
- Object storage
 - each file stored is a discrete object: no access to parts of files
 - storage and retrieval of objects only

Off-site backup for raw data: examples

- Sequence repositories like SRA or ENA
 - free, and usually required anyway, BUT ...
- Cloud object storage
 - e.g., Amazon S3 Glacial (¢ but € to retrieve)
- Cloud filesystems or central file servers
 - Dropbox, Box, mounted remote drives
 - Feature-rich and expensive : better for scripts and results
- Attached block filesystem with tape backup
 - Regular working storage, or something like
 - UPPMAX's lutra (<https://www.uppmax.uu.se/uppmax-news/?tarContentId=814696>)
- Individual hard drives or tape drive + tapes
 - Very common and often the only suitable option

Object storage

Object storage

Block storage

Block storage

Hybrid storage

Hybrid storage

Changes made by ENA to FastQ files

- Sequence identifier: `accession.read-number`
 - read-number is sequential from the start of the file
- Description: `read-number/read-of-pair`
- Quality values may be re-scaled
 - https://en.wikipedia.org/wiki/FASTQ_format

```
$ zcat SRR10794580_1.40000.fastq.gz | head -n 16
@SRR10794580.1 1/1
NATATTAACAAGCTAATTTACATATTCAGATAACCCTAACTCTAACCTA
+
#FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SRR10794580.2 2/1
AAAGTTGATAGGGCAGACGTTCTGAATGGGTCGTCGCCGCCACGGGGGGC
+
FFFF:FFFFFFFFFF:FFFFFFFFFF:FFFFFFFFFFFFFFFFFFFFFFFF
@SRR10794580.3 3/1
AAAGTTGATAGGGCAGACGTTCTGAATGGGTCGTCGCCGCCACGGGGGGC
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFF,FFFFFFFFFFFFFFFFFFFFFFFF
@SRR10794580.4 4/1
CTCCTACTTGATAACTGTGGTAATTCTAGAGCTAATACATGCCGACGG
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

```
$ zcat SRR10794580_2.40000.fastq.gz | head -n 16
@SRR10794580.1 1/2
GACAGCATCTCTCTCCTTGTTGCACAGGCTGGAGGGGTGGTGCGATCAC
+
F:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SRR10794580.2 2/2
GTACATGGGTACCTGGTTGATCCTGCCAGTAGCATATTGCTTGTCTCAA
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SRR10794580.3 3/2
GTACATGGGTACCTGGTTGATCCTGCCAGTAGCATATTGCTTGTCTCAA
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SRR10794580.4 4/2
GTCCTGTATTGTTATTTTCGTCACTACCTCCCCGGGTCGGGAGTGGGT
+
FFFFFFF:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

Off-site backup for other than raw data

- Small-ish size
- High frequency of access
 - Both upload and download frequently
- Version control highly desired
- Available on local block storage filesystem
- Off-site mirror may be object storage or something else
- Cloud filesystems or central file servers can work well
- Git repositories with remotes are ideal
 - May keep remote repositories private while working
 - Make public when released/published
 - Large file support is a desirable add-on
- Ad hoc solutions also work, but can easily become neglected

Github gives
free private
repositories
to academics

Transfers to and from off-site backup

- Raw data must have checksums (more later!)
- Transfers must ensure for data integrity
 - Uploads to sequence repositories must include checksums
- Repositories may suggest or require specific upload tools
 - Git `git`, Amazon `awscli`, Google's Cloud SDK (`gsutil`, `gcloud`)
 - May suggest, e.g., IBM Aspera or its `ascp`, but often not required
- `rsync -Pa` is ideal
 - secure transfers with continuation and preserved file attributes
 - during transfers, checksums are computed on both ends and compared
- `lftp`, `scp`, `sftp`, `ftp` if no other option (`lftp` is best)
 - these only transfer data (securely if the endpoints support that)
 - data integrity must be ensured some other way (checksums!)

Collecting multiple files into one archive: tar

- Old-school format, name comes from Tape ARchive
- Options specify operation, compression, etc.
- Can contain files or complete directory trees

- List contents of a tar archive: `tar -t`

```
tar -tvf Scofield.tar.gz
  -t          table of contents
  v          verbose operation
  f tar-file contents of tar-file
```

f is a 'sticky' option, the name of the tar file must immediately follow it

- Modern tar detect compression automatically, but

```
tar -tvzf Scofield.tar.gz    z  gzip compression
tar -tvjf Scofield.tar.bz2   j  bzip2 compression
tar -tvJf Scofield.tar.xz    J  xz compression
```

Extract contents from a tar archive

- Can extract all files, or a selection
- Compression methods are detected automatically
- Extract contents of a tar archive: `tar -x`

```
tar -xvf Scofield.tar.gz
    -x          extract contents
    v          verbose operation
    f tar-file contents of tar-file
```

this extracts the complete archive
contents to the current directory

- Extract specific contents: file or complete subdirectory tree

```
tar -xvf Scofield.tar.gz file-within-archive
tar -xvf Scofield.tar.gz directory-within-archive
```

- Extract contents into a specific directory

```
tar -xvf Scofield.tar.gz -C some-directory
```

Create a tar archive

- Specify files or directories
- Must specify compression method, if desired
- Create a tar archive: `tar -c`

```
tar -cvf Scofield.tar.gz directory1 file1 ...
  -c                create archive
  v                verbose operation
  j                bzip2 compression
  f tar-file      name of new tar-file
  ...              contents of new tar archive
```

- Updating a tar archive (`tar -u`) is not recommended
 - Adds new versions to the end, it does not replace
 - Instead, create a new archive

A quick note on compression

- Repeated sequences are replaced with shorter ones
- A table of repeated sequences is maintained and refreshed
- A compressed block includes this table plus compressed data
- Compression formats may use
 - one block per complete file gzip
 - multiple blocks per file (e.g., every 100 KB) bzip2, xz, ...
- An error within a compressed block usually invalidates the remainder of the compressed block, so
 - One block per file: the remainder of the file is lost (!!!!!)
 - Multiple blocks: only the remainder of the block is lost
- Which is better for archival use?

bgzip (used for BAM, indexed VCF, etc.) is a blocked version of gzip

Why is a BAM file of mapped reads quite a bit smaller than a BAM file of unmapped reads?

Checksums (finally)

- A small number calculated by scanning through data
 - Calculate checksums and include them with data
 - Receiver recalculates and compares checksums
- Different methods for calculating checksums
 - md5, SHA-1, SHA-2 (sha224, sha256, sha384, sha512), SHA-3, ...
 - checksum with additional assurances: pgp, certificates, ...
- Checksums ensure data integrity: you have what I have
- Do not necessarily ensure data security (e.g, tamper-proof)
 - md5 and sha-1 are "broken": can create file with arbitrary checksum
 - but both are still very widely used
 - others are not close to being broken (SHA-2 underlies Bitcoin)

A small difference in the data results in a big difference in the checksum

Verifying checksums

- Data with checksum, here md5

```
-rw-rw-r-- 1 dousc151 staff 5333131 May 11 16:04 Scofield.tar.gz
-rw-rw-r-- 1 dousc151 staff      50 May 11 16:10 Scofield.tar.gz.md5
$ cat Scofield.tar.gz.md5
11eb13f44285bd21ae8f39e759ee59b2 Scofield.tar.gz
```

- Recompute checksum manually and compare:

```
$ md5sum Scofield.tar.gz
11eb13f44285bd21ae8f39e759ee59b2 Scofield.tar.gz
```

- If the md5 file is output from md5sum, check directly with -c:

```
$ md5sum -c Scofield.tar.gz.md5
Scofield.tar.gz: OK
```

Note here we use
the .md5 file itself

Computing checksums

```
md5sum file1 file2 ... > files.md5  
sha256sum file1 file2 ... > files.sha256
```

- Checksums are calculated on files, not directories
 - `md5sum directory` is an error
- An `.md5` files can contain checksums for multiple files
 - `md5sum -c file.md5` will check each of them
- Example:

```

$ tar -xf Scofield.tar.gz
$ ls -l
total 5224
drwxrwxr-x 6 dousc151 staff 192 May 11 13:33 SRR10794580
drwxrwxr-x 6 dousc151 staff 192 May 11 13:33 SRR2292852
drwxrwxr-x 6 dousc151 staff 192 May 11 13:33 SRR2989017
drwxrwxr-x 5 dousc151 staff 160 May 11 13:33 SRR609851
drwxrwxr-x 5 dousc151 staff 160 May 11 13:33 SRR609888
-rw-rw-r-- 1 dousc151 staff 5333131 May 11 16:04 Scofield.tar.gz
-rw-rw-r-- 1 dousc151 staff 50 May 11 16:10 Scofield.tar.gz.md5
-rw-rw-r-- 1 dousc151 staff 1350 May 11 13:37 data.md5
-rwxrwxr-x 1 dousc151 staff 2644 May 11 13:33 fetch_data.sh
$ cat data.md5
71750d875901debdcca18a7a3a515d67 ./SRR609851/SRR609851.report.tsv
bc548f4af7a94e69140c60c13b2edf73 ./SRR609851/SRR609851.txt
e7df943146ac001fcaa4a2d80cb74df3 ./SRR609851/SRR609851.40000.fastq.gz
a9d89d7efb5760b85b70a75e0410d6da ./SRR609888/SRR609888.report.tsv
5a54303449278c5eecf2bdd01aaf602b ./SRR609888/SRR609888.txt
3184365f165e10af4ebafba1538ac119 ./SRR609888/SRR609888.40000.fastq.gz
6f696f2fc7fae0be16deb95fc9d311c7 ./SRR2989017/SRR2989017.report.tsv
d81ce948df6483e076553fd8c6f8e878 ./SRR2989017/SRR2989017.txt
4a94348d96c11bf41cfd3df75f4ac325 ./SRR2989017/SRR2989017_1.40000.fastq.gz
e6cd7aed849315c775cdae3bcde71b4d ./SRR2989017/SRR2989017_2.40000.fastq.gz
0587e4a3b668e63f49d4fe7fc9f5e2b2 ./SRR10794580/SRR10794580.report.tsv
6f8ed785065d72438872269b9a216c16 ./SRR10794580/SRR10794580.txt
33490e4474ff071a5e4bde7330e8f230 ./SRR10794580/SRR10794580_1.40000.fastq.gz
848c3c6f6d53623292c2f71cab81c97c ./SRR10794580/SRR10794580_2.40000.fastq.gz
c7dc6f608accf041d0eecf3a7e6b23e4 ./SRR2292852/SRR2292852.report.tsv
c87f3eeb2b718d7cce1a62d448565a19 ./SRR2292852/SRR2292852.txt
07613e02057b904fb1f6341a43c9f721 ./SRR2292852/SRR2292852_1.40000.fastq.gz
2dc18dd5a1eba7628ae35fd78ae41228 ./SRR2292852/SRR2292852_2.40000.fastq.gz
d41d8cd98f00b204e9800998ecf8427e ./data.md5
85fafb76391e30217292cc205e4955e6 ./fetch_data.sh

```

```

$ md5sum -c data.md5
./SRR609851/SRR609851.report.tsv: OK
./SRR609851/SRR609851.txt: OK
./SRR609851/SRR609851.40000.fastq.gz: OK
./SRR609888/SRR609888.report.tsv: OK
./SRR609888/SRR609888.txt: OK
./SRR609888/SRR609888.40000.fastq.gz: OK
./SRR2989017/SRR2989017.report.tsv: OK
./SRR2989017/SRR2989017.txt: OK
./SRR2989017/SRR2989017_1.40000.fastq.gz: OK
./SRR2989017/SRR2989017_2.40000.fastq.gz: OK
./SRR10794580/SRR10794580.report.tsv: OK
./SRR10794580/SRR10794580.txt: OK
./SRR10794580/SRR10794580_1.40000.fastq.gz: OK
./SRR10794580/SRR10794580_2.40000.fastq.gz: OK
./SRR2292852/SRR2292852.report.tsv: OK
./SRR2292852/SRR2292852.txt: OK
./SRR2292852/SRR2292852_1.40000.fastq.gz: OK
./SRR2292852/SRR2292852_2.40000.fastq.gz: OK
./data.md5: FAILED
./fetch_data.sh: OK
md5sum: WARNING: 1 computed checksum did NOT match

```

**data.md5 failed because it was
in the midst of being created
when its md5 was calculated**

Computing multiple checksums

- `md5deep -r` does this for a directory tree

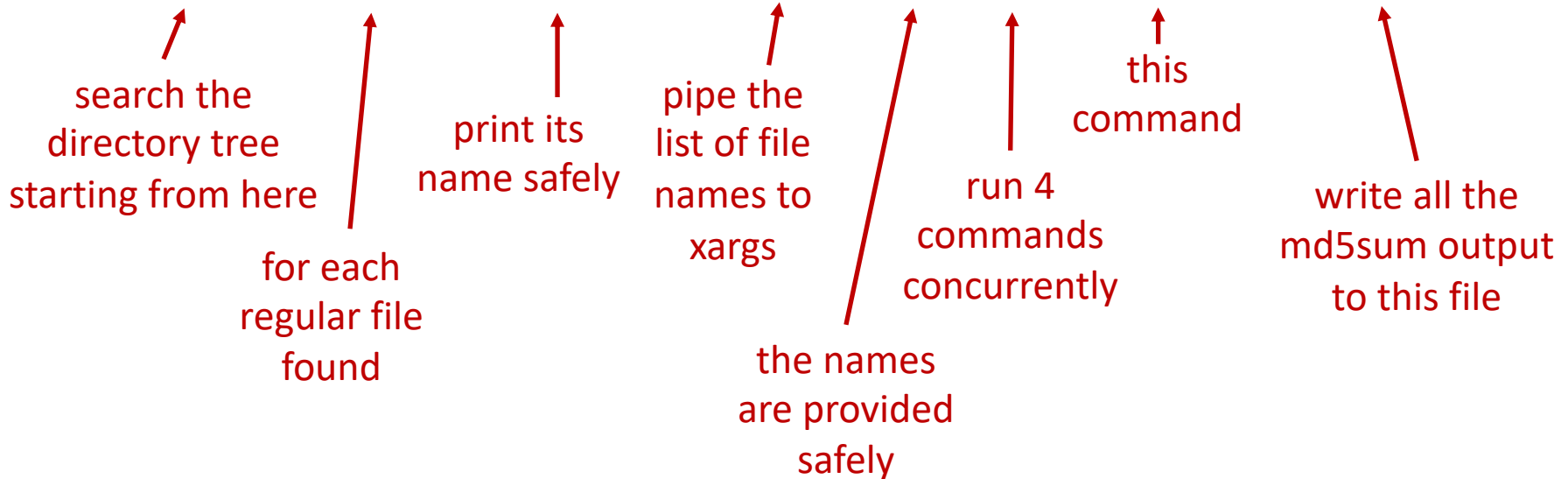
```
$ md5deep -r SRR609888
5a54303449278c5eecf2bdd01aaf602b /domus/h1/douglas/Cesky/checksums/SRR609888/SRR609888.txt
a9d89d7efb5760b85b70a75e0410d6da /domus/h1/douglas/Cesky/checksums/SRR609888/SRR609888.report.tsv
3184365f165e10af4ebafba1538ac119 /domus/h1/douglas/Cesky/checksums/SRR609888/SRR609888.40000.fastq.gz
$ md5deep -r -l SRR609888
5a54303449278c5eecf2bdd01aaf602b SRR609888/SRR609888.txt
a9d89d7efb5760b85b70a75e0410d6da SRR609888/SRR609888.report.tsv
3184365f165e10af4ebafba1538ac119 SRR609888/SRR609888.40000.fastq.gz
```

- `md5deep -r -l` outputs relative directory paths
- You may not have or may not be able to install `md5deep`
- Linux tools to the rescue :-D

Computing checksums in parallel with xargs

- `find -print0` goes together with `xargs -0`
- `-print0` with `-0` ensures spaces in filenames handled correctly

```
find . -type f -print0 | xargs -0 -P 4 md5sum > data.md5
```



Computing checksums in parallel with parallel

- `find -print0` goes together with `parallel -0`
- Here a `sort` step ensures the output is sorted by filename

```
find . -type f -print0 | parallel -0 -j 4 md5sum | sort -k2,2V > ...
```

search the directory tree starting from here

for each regular file found

print its name safely

pipe the list of file names to parallel

the names are provided safely

run 4 commands concurrently

this command

pipe all the md5sum output to sort

sort the output on the second field, the filename, using Version sort

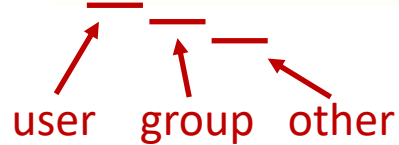
`parallel` has a `--dry-run` option and many other useful features

Managing access for a project

- Linux file and directory permissions are important
- So are Linux user ('owner') and group identities
- Separate projects should have separate Linux groups
 - Many compute centres separate projects using Linux groups
 - If you manage compute resources, you should do the same

permissions user group project directory

```
drwxrwsr-x 2 douglas sllstore2017033 4096 May 19 17:56 flycatcher_comparison
```



The diagram shows three red arrows pointing from the permission string 'drwxrwsr-x' to the labels 'user', 'group', and 'other' below it. The first arrow points from the 'd' to 'user', the second from the first 'r' to 'group', and the third from the second 'r' to 'other'.

- only the user/owner can change a file's attributes
 - but, depending on permissions, anyone can modify or delete files

Keeping things consistent for project members

- Use 'set group ID' on the directories

- `chmod g+s top-level-directory`

Use only on
directories!!!!

```
rackham5: ~/Cesky/projects $ ls -ld flycatcher_comparison
drwxrwxr-x 2 douglas sllstore2017033 4096 May 19 17:56 flycatcher_comparison
rackham5: ~/Cesky/projects $ chmod g+s flycatcher_comparison
rackham5: ~/Cesky/projects $ ls -ld flycatcher_comparison
drwxrwsr-x 2 douglas sllstore2017033 4096 May 19 17:56 flycatcher_comparison
```

- With this, new directories/files inherit parent directory group
 - With the group set correctly, group permissions work correctly
- Users should set `umask 0002` in `.profile`, `.bashrc`, `.zshrc`, ...
 - Files and directories are created with 'user' and 'group' write permissions and 'other' with no write permission
 - Since group == project, this means all project members can modify files and directories created by other project members

Linux default is `0022`, user/owner can write, group members and others cannot write

Applying group IDs and 'set group ID'

Two steps to apply to the whole project directory:

- Change group IDs for all files and directories

```
chgrp -R group project-directory
```

- Set 'set group ID' on all directories, and only directories

```
find project-directory -type d -exec chmod g+s {} \;
```

search the
directory tree
starting from here

for each directory
found, including
the top

run this
command

insert the
directory
name

end of the
command

Groups keep things tidy

- Without this, group IDs can get messy ...

drwxrwxr-x	2	douglas	douglas	4096	Aug	3	2017	cow
-rw-rw-r--	1	douglas	douglas	1310	May	25	2018	cronta
-rwxrwxr-x	1	douglas	b2012190	1042	Dec	5	2014	cutada
-rw-rw-r--	1	douglas	snic2015-6-172	2079	Jan	8	2016	cutada
-r--r--r--	1	douglas	b2012190	3163	Dec	5	2014	cutada
-rw-rw----	1	douglas	b2012190	10673	Dec	5	2014	cutada
-rwxrwxr-x	1	douglas	b2012190	2866	Mar	7	2019	detect
-rwxrwxr-x	1	douglas	sw	398	Sep	2	2019	diskus
-rw-rw-r--	1	douglas	douglas	401	Oct	5	2022	double

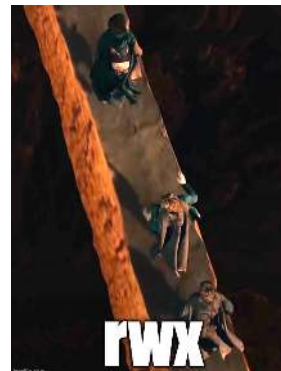
- Even with this, the group will not be changed for:
 - files that have been moved with `mv`
 - files that have been copied with `cp -p` or `cp -a`
- The user/owner can still change the group to something else

Excluding 'others': not group members

- 'Other' users should not have write permission
 - `chmod -R o-w directory` to restrict a directory tree
 - If every user has `umask 0002` this becomes the default
- If the desire for privacy is high, give 'others' no permissions
 - `chmod o-rwx directory`

```
rackham5: ~/Cesky/projects $ chmod o-rwx flycatcher_comparison
rackham5: ~/Cesky/projects $ ls -ld flycatcher_comparison
drwxrws--- 2 douglas sllstore2017033 4096 May 19 17:56 flycatcher_comparison
```

- This only needs to be done to the top directory in the tree



Raw data

- Identify all the raw data under your or your group's control
 - Delivered sequences **BUT ALSO**
 - Slide scans, imaging, chemostat logs, ...
 - Field sheets/notebooks, lab notes/notebooks, ...
 - Raw data includes its metadata!
- Anything that is not digital, create a digital version
 - Scan as PDF or take pictures
 - Make it a practice (back at camp, every Friday, ...)
- Intermediates are important, but not substitutes!
 - e.g., an Excel sheet containing transcribed/corrected field data
 - Intermediates can be recreated from raw data

Tiring, but not as
tiring as losing
your data

Tiring, but not as
tiring as losing
your data

Metadata: data about data

- The other information you have about your raw data
- Sample IDs, locations, tissue of origin, collection methods, ...
- Metadata may refer to other metadata
 - E.g, a sample ID can be an 'index' into a sample information table
 - Citations to methods, citations to publications using these data, etc.
- Also, checksums for data integrity (more later)
- Document methods for making use of metadata
- Make metadata obvious

METADATA.md README.txt READ_HERE_FIRST/

- More is better

The people you are most helping with rich metadata
are you, your group, and your collaborators

Managing raw data

- Clearly delineate raw data from all other materials
 - Segregate raw data in a separate top-level directory tree
 - Alternate views can also be created
- Place metadata alongside the raw data
 - Sequencing delivery reports are not sufficient!!
 - For raw data in particular, metadata should be 'self-contained'
 - The ideal: enough detail to reproduce these data
- Write-protect and delete-protect raw data and metadata
 - write-protect a file: `chmod -w file`
 - delete-protect contents of a directory: `chmod -w directory`
- Back up raw data and metadata off-site as soon as possible
 - suitability of backup schemes may be determined by regulations

More on this later

Write- and delete-protecting files

- write-protect a file: `chmod -w file`
- delete-protect directory contents: `chmod -w directory`

```
rackham5: ~/Cesky/SRR10794580 $ ll
total 780
-rw-rw-r-- 1 douglas staff    393 May 11 13:33 SRR10794580.report.tsv
-rw-rw-r-- 1 douglas staff   2128 May 11 12:29 SRR10794580.txt
-rw-rw-r-- 1 douglas staff 377328 May 11 13:33 SRR10794580_1.40000.fastq.gz
-rw-rw-r-- 1 douglas staff 399591 May 11 13:33 SRR10794580_2.40000.fastq.gz
rackham5: ~/Cesky/SRR10794580 $ chmod -w *
rackham5: ~/Cesky/SRR10794580 $ echo "add this" >> SRR10794580_1.40000.fastq.gz
-bash: SRR10794580_1.40000.fastq.gz: Permission denied
rackham5: ~/Cesky/SRR10794580 $ rm -f SRR10794580_1.40000.fastq.gz
rackham5: ~/Cesky/SRR10794580 $ chmod -w .
rackham5: ~/Cesky/SRR10794580 $ rm -f SRR10794580_2.40000.fastq.gz
rm: cannot remove 'SRR10794580_2.40000.fastq.gz': Permission denied
```

- write- and delete-protect directory tree:
`chmod -R -w directory`

-R : recursive

What about external raw data?

- E.g, files from ENA/SRA, or produced by collaborators
- These are raw data but not your raw data. Consider whether
 - Backup is unnecessary (ENA/SRA)
 - Unless downloading is expensive in time/effort/bandwidth
 - Backup is not your responsibility (collaborators)
 - Unless pre-arranged
 - There may also be some regulatory guidance
- Distinguish clearly from other raw data
 - separate directory trees with clear naming and metadata
 - external data that is not backed up
 - external data that is backed up

Organizing raw data ... my suggestion

- First, by data source

```
RAW_DATA/  
  DELIVERIES/  
    Delivery0984478/  
      delivery reports and other raw data  
      sequence files  
    Delivery34_XJ100/  
      ...  
  FIELD_STUDIES/  
    Summer2023/  
      scanned worksheets  
      ...
```

Identify the relevant files you want
to access from the project

- Can be placed with a minimum of fuss
- Immediately protected and backed up

Create a *view* using folders and symlinks

- A lightweight alternate representation of relevant data

```
RAW_DATA/  
  BY_SAMPLE/  
    x001/
```

The view is where we
“expose” relevant files

```
      symlinks to sequence files for sample x001  
  y028/  
      symlinks to sequence files for sample y028  
  ...
```

- More direct access to relevant files
- Symbolic links are self-documenting
- Permissions still determined by the original files
- Link using absolute paths
- Best to script the creation of a view

Use symbolic links to clear things up

Do not create
hard links: `ln`
without `-s` !!!

`ln -s original-file symlink-file`

```
SAMPLE $ ls -l
```

A symbolic link shows what it refers to

```
sample1.r1.fastq.gz -> /Users/dousc151/Dropbox/Cesky/SRR10794580/SRR10794580_1.40000.fastq.gz
sample1.r2.fastq.gz -> /Users/dousc151/Dropbox/Cesky/SRR10794580/SRR10794580_2.40000.fastq.gz
sample2.r1.fastq.gz -> /Users/dousc151/Dropbox/Cesky/SRR22292852/SRR22292852_1.40000.fastq.gz
sample2.r2.fastq.gz -> /Users/dousc151/Dropbox/Cesky/SRR22292852/SRR22292852_2.40000.fastq.gz
sample3.r1.fastq.gz -> /Users/dousc151/Dropbox/Cesky/SRR2989017/SRR2989017_1.40000.fastq.gz
sample3.r2.fastq.gz -> /Users/dousc151/Dropbox/Cesky/SRR2989017/SRR2989017_2.40000.fastq.gz
sample4.fastq.gz -> /Users/dousc151/Dropbox/Cesky/SRR609851/SRR609851.40000.fastq.gz
sample5.fastq.gz -> /Users/dousc151/Dropbox/Cesky/SRR609888/SRR609888.40000.fastq.gz
```

```
UU-FVFGJ2H2Q05N: ~/Dropbox/Cesky/BY_SAMPLE $ ls -lL
```

```
total 5280
```

```
-rw-r--r-- 1 dousc151 377328 May 11 13:33 sample1.r1.fastq.gz
-rw-r--r-- 1 dousc151 399591 May 11 13:33 sample1.r2.fastq.gz
-rw-r--r-- 1 dousc151 628390 May 11 13:33 sample2.r1.fastq.gz
-rw-r--r-- 1 dousc151 650810 May 11 13:33 sample2.r2.fastq.gz
-rw-r--r-- 1 dousc151 1115401 May 11 13:33 sample3.r1.fastq.gz
-rw-r--r-- 1 dousc151 1133540 May 11 13:33 sample3.r2.fastq.gz
-rw-r--r-- 1 dousc151 838102 May 11 13:33 sample4.fastq.gz
-rw-r--r-- 1 dousc151 246035 May 11 13:33 sample5.fastq.gz
```

`ls -L` instead shows
the symlink file with
the attributes of the
original file

Script the creation of a view

```
#!/usr/bin/env bash

RAW=/Users/dousc151/Dropbox/Cesky

BY_SAMPLE=$RAW/BY_SAMPLE

mkdir -p $BY_SAMPLE
cd $BY_SAMPLE

ln -s $RAW/SRR10794580/SRR10794580_1.40000.fastq.gz sample1.r1.fastq.gz
ln -s $RAW/SRR10794580/SRR10794580_2.40000.fastq.gz sample1.r2.fastq.gz
ln -s $RAW/SRR22292852/SRR22292852_1.40000.fastq.gz sample2.r1.fastq.gz
ln -s $RAW/SRR22292852/SRR22292852_2.40000.fastq.gz sample2.r2.fastq.gz
ln -s $RAW/SRR2989017/SRR2989017_1.40000.fastq.gz sample3.r1.fastq.gz
ln -s $RAW/SRR2989017/SRR2989017_2.40000.fastq.gz sample3.r2.fastq.gz
ln -s $RAW/SRR609851/SRR609851.40000.fastq.gz sample4.fastq.gz
ln -s $RAW/SRR609888/SRR609888.40000.fastq.gz sample5.fastq.gz
```

- All the benefits of a script: repeatable, editable, and so on
- Simplifies the use of absolute paths
- A slightly extended script could create a view anywhere

A script: What, and why?

- A script is a file containing statements to be interpreted
- A Bash script contains statements for the Bash shell
 - familiar commands (grep, cat, etc.)
 - Bash syntax you are learning (< , > , | , \$(...), \${...}, etc.)
 - Bash syntax for control flow (&& , || , if , for , & , wait , etc.)
 - Comments (lines that start with #)
- Python scripts, Perl scripts, etc.
- A script both describes and performs some process
 - it can be viewed without interpreting (“running”) it
- A script’s behaviour can be modified using parameters
- A script can be reused, by you or someone else

The slightly extended script to create a view

```
#!/usr/bin/env bash

# if no argument, creates and populates BY_SAMPLE under the RAW directory
# if one argument, creates and populates BY_SAMPLE under that directory

RAW=/Users/dousc151/Dropbox/Cesky # top-level directory

if [[ $# == 0 ]] ; then    # no argument given
    BASE="$RAW"
elif [[ $# != 1 ]] ; then
    echo "$0: only 0 or 1 argument allowed"
    exit 1
else # create links at the directory named by the argument
    BASE="$1"
fi

BY_SAMPLE=$BASE/BY_SAMPLE

mkdir -p $BY_SAMPLE
cd $BY_SAMPLE

ln -s $RAW/SRR10794580/SRR10794580_1.40000.fastq.gz sample1.r1.fastq.gz
ln -s $RAW/SRR10794580/SRR10794580_2.40000.fastq.gz sample1.r2.fastq.gz
ln -s $RAW/SRR22292852/SRR22292852_1.40000.fastq.gz sample2.r1.fastq.gz
ln -s $RAW/SRR22292852/SRR22292852_2.40000.fastq.gz sample2.r2.fastq.gz
ln -s $RAW/SRR2989017/SRR2989017_1.40000.fastq.gz sample3.r1.fastq.gz
ln -s $RAW/SRR2989017/SRR2989017_2.40000.fastq.gz sample3.r2.fastq.gz
ln -s $RAW/SRR609851/SRR609851.40000.fastq.gz sample4.fastq.gz
ln -s $RAW/SRR609888/SRR609888.40000.fastq.gz sample5.fastq.gz
```

Organizing the rest of the project

- The details vary a lot. Project structure may be
 - imposed: this is the way
 - inherited: this was the way
 - free-for-all: there is no way
 - inefficient: what is the way?
- Apart from segregating all raw data, much is flexible
- A good project structure
 - makes different types of data easy to find (BAMs, VCFs, genomes, ...)
 - connects data to project-relevant ontologies (samples, subprojects, ...)
 - avoids duplication, especially inadvertant duplication
 - is functional: supports asking, developing, and expanding hypotheses

Top-level project organization

1 through 6 are the types of data a group should have conversations about

1. Raw data
2. External raw data
3. External reference data (genome, annotation, etc.)
 - Create project-local symlinks If using local mirrors (e.g., iGenomes)
4. Project products: reference data (draft, final, ...)
5. Project products: derived data (VCFs, expression tables)
6. Project products: important intermediates (refined BAMs, ...)
7. Subdirectories for users and subprojects: “everything else”
 - repeat discovery
 - SV pipelines
 - experimental projects
 - ...

Organizing “everything else”

7. Subdirectories for users and subprojects: “everything else”

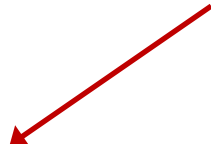
- No new raw data!
- Scripts, tools, conda environments, containers, databases, ...
- Off-site backup of scripts and other important stuff
- Organize around
 - producing and comparing derived products
 - answering questions via comparison and exploration
- Liberal use of localized documentation and metadata
 - NOTES.md, METHODS.md, CrazyThoughts.txt
- The person you are most likely to be communicating with is yourself and your immediate collaborators, in the near future

all the beautiful things that happen during
biological research

Bash (shell) job control

- Typically a command is running in the foreground
 - the shell waits for it to complete before returning a prompt
- Commands can be run in the background by appending ‘&’
 - useful if the command might take a while to complete

```
$ find RAW_DATA/ -type f -name '*.fastq.gz' -exec md5sum {} \; > checksums.md5 &  
[1] 15621  
$  
[1]+  Done                  find RAW_DATA/ -type f -name '*.fastq.gz' -exec md5sum {} \; > checksums.md5  
$ head -n 4 checksums.md5  
3184365f165e10af4ebafba1538ac119 RAW_DATA/SRR609888/SRR609888.40000.fastq.gz  
e7df943146ac001fcaa4a2d80cb74df3 RAW_DATA/SRR609851/SRR609851.40000.fastq.gz  
4a94348d96c11bf41cfd3df75f4ac325 RAW_DATA/SRR2989017/SRR2989017_1.40000.fastq.gz  
e6cd7aed849315c775cdac3bcde71b4d RAW_DATA/SRR2989017/SRR2989017_2.40000.fastq.gz
```



[1]	job control job number
15621	process ID (PID) of the job
[1]+	job number with relative job position

a job can be ‘Stopped’: it is in the background, but is not running

Job numbers and relative job numbers

- [1], [2], etc. are assigned as background jobs are created
- + is assigned to the most recent job, - to the next most recent
- Can refer to jobs using %1, %2, %+ , %- or using process ID

```
$ sleep 500 &
[1] 15859
$ sleep 600 &
[2] 15860
$ sleep 10 &
[3] 15861
$ jobs
[1]    Running
[2]-   Running
[3]+   Running
$
[3]+   Done
$ jobs
[1]-   Running
[2]+   Running
```

relative job
numbers change
as jobs end

```
sleep 500 &
sleep 600 &
sleep 10 &
sleep 10
sleep 500 &
sleep 600 &
```

```
$ sleep 700 &
[3] 16055
$ jobs
[1]    Running
[2]-   Running
[3]+   Running
```

```
$ kill %1
$
[1]    Terminated: 15
$ kill %+
$
[3]+   Terminated: 15
$ jobs
[2]+   Running
```

naming jobs
using %

```
sleep 500 &
sleep 600 &
sleep 700 &
sleep 500
sleep 700
sleep 600 &
```

Uses of shell job control

- Multiple commands can be run in the background
- Useful within a script, too, including a batch script, **BUT**
- Use wait to wait until all background processes are done
 - Say, if background processes are creating files needed for a next step
 - Or the script reached its end
- without `wait`, a script may finish before its background processes
 - with many batch systems, this will kill all processes run by the batch job including the background processes
 - So, if you use job control in scripts (batch or otherwise), also use `wait`
- if you want to run many similar tasks, a few at a time, don't use job control, use `parallel`

Job control commands

- `&` Put new process in the background immediately
 - `jobs` List background processes
 - `Ctrl-c` Kill the foreground process (job is ended)
 - `Ctrl-z` Stop the foreground process (job is Stopped)
 - `bg` Continue running stopped process but in background
 - `fg` Move background process to foreground
-
- By default, `bg` and `fg` affect `%+`, the most recently job
 - But, any of the `%` names can be used

play around with long sleep and these commands

Running within a screen

- A screen contains shell that can persist until the next reboot

- Create a screen, with a name

`screen -S transfer`

unless you exit the shell !!

- List your screens; this one is now Attached

`screen -ls`

a terminal is
interacting with it

- Run something, like a long sleep
- Disconnect from the screen; the prompt is not needed!

`Ctrl-a d`

- List your screens; this one is now Detached

`screen -ls`

a terminal is not
interacting with it

- Reconnect to the Detached screen using its name

`screen -R transfer`

screen use cases

- Essential when logging in over a poor connection
- Essential when running multiday tasks (e.g., downloading data)
- Workflows: one screen per 'long' task at the command line
 - Instead of separate windows, separate screens ?
 - Or some mix
- Screens exist on the computer they were started on
 - Can only reconnect when on the same computer
 - E.g., start a screen in office, reconnect from home
 - `screen -R` transfer only reattaches a Detached screen
 - `screen -R -DD` transfer will 'steal' a screen
 - Detach, then attach

If the
screen is
Attached, it
starts a
new screen

screen has many other capabilities

- start a screen within a screen
 - it is doable, but avoid doing this (usually done by mistake)
- Logging all input and output (avoid doing this, too)
- Start a command within a new screen
 - `screen vim script.sh`
 - the screen has no shell: only this command
- `screen -ls` shows the ‘full’ name of each screen
 - `screen -R` will work using a minimally unique substring

```
rackham5: /domus/h1/douglas $ screen -ls
There are screens on:
      37331.working_big.download.lftp (Detached)
      19189.1000_genomes_project.lftp (Detached)
2 Sockets in /var/run/screen/S-douglas.
```

- Many, many other capabilities: `man screen` for more

Thanks!

- Questions
- Ponderings
- Curious problems
- ... ?